

The classpack \LaTeX 2_ε package*

XML mastering for \LaTeX classes and packages

Literate-programming solution
for class and package maintenance

Peter Flynn
Silmaril Consultants
Textual Therapy Division
(peter@silmaril.ie)

26th February 2024

Summary

\LaTeX document classes and packages are conventionally created, maintained, and distributed in Doc \TeX (.dtx) format using the ltxdoc class, which provides for interleaved code and documentation (‘literate programming’). However, the accurate construction of these files is technically challenging, and editing them is tedious and error-prone.

ClassPack allows a \LaTeX developer to create a *DocBook5* XML document for a class or package, containing the code interleaved with annotations, plus a section for the documentation. It provides an xslt3 script to generate the .dtx, .ins, and other files, which can be suitably zipped up for submission to CTAN.

This is a development system for class and package authors. The classpack.sty file contains typographic adjustments and utilities only needed to typeset the documentation of *ClassPack*-produced classes or packages.

This software is a work-in-progress. It has been in development at Silmaril since 1998, and in production use since 2008. This release includes completion of the rewrite of the *AutoPackage* algorithms, improved Makefile control, and a detailed update of the annotation of the xslt code. A paper describing an earlier version was presented at a Balisage markup conference in Montréal (Flynn, 2013).

○	<title>XML mastering for &LaTeX; classes and packages</title>
○	<subtitle>Literate-programming solution <?LaTeX \\\?>
○	for class and package maintenance</subtitle>
○	<author>
○	<personname>
○	<firstname>Peter</firstname>
○	<surname>Flynn</surname>

*This document corresponds to classpack v. 1.28, dated 2024/02/21.

Contents

1	Summary of the <i>ClassPack</i> code management system	10
1.1	Code management features in <i>ClassPack</i>	10
1.2	Presentation features in the classpack package itself	11
1.2.1	Features adjusted for ease of reading	11
1.2.2	Features adjusted for ease of writing	12
2	Introduction	13
2.1	XML	13
2.2	<i>ClassPack</i>	13
2.2.1	Use of XML	15
2.2.2	Changes to the DocBook DTD	16
3	Setting up a <i>ClassPack</i> document	18
3.1	XML editing	19
3.2	<i>ClassPack</i> document structure	20
3.3	Configuration for a new class or package	21
3.4	Metadata	24
3.5	Part "doc": the user documentation	26
3.5.1	Hierarchical structure	26
3.5.2	Structural markup	27
3.5.3	Inline markup	27
3.6	Part "code": the annotated code	27
3.6.1	Annotations	27
3.6.1.1	Specifying colours	29
3.6.1.2	Reusability	29
3.6.2	Ancillary annotated files	30
3.7	Part "files": unannotated ancillary files	30
3.8	The README.md file	31
3.9	Other output	32
3.10	Worked example	32
3.10.1	Startup	32
3.10.2	Root element specifications	33
3.10.3	Packages for the documentation	33
3.10.3.1	Ancillary startup commands for the documentation	33
3.10.4	Packages for the class or style package	34
3.10.5	The Manifest	35
3.10.6	Metadata	35
3.10.7	Documentation	36
3.10.8	Code	37
3.10.9	Extractable <i>annotated</i> files	38
3.10.10	Extractable files without annotation	38
4	Package management	39
4.1	Where to put the lists of packages	39
4.2	Listing the packages	40

4.3	Specifying where in the .dtx file to output packages for a class or package	44
4.4	Additional setup commands for documentation	45
4.4.1	LaTeX commands to appear in the documentation Preamble	45
4.4.2	LaTeX commands to appear before the start of the documentation text	45
4.5	The Manifest	46
5	Automated package inclusion	48
5.1	Packages preloaded before the document class declaration	48
5.2	The prepost.xml lookup file	49
5.3	Specifying known packages and default options	51
5.4	Automating inclusion	52
5.4.1	Default inclusion (unconditional)	53
5.4.2	Inclusion when a particular element type is used	53
5.4.3	Inclusion when an attribute on an element type is used	53
5.4.4	Inclusion when an attribute on an element type has a specific value	53
5.4.5	Inclusion when an element references a specific element type via the ID/IDREF mechanism	54
5.4.6	Restricting the loading of conflicting packages	54
5.5	Adding setup commands before or after a package	55
6	Using <i>ClassPack</i> to write the documentation	56
6.1	Writing assistance in XSLT	56
6.2	Hierarchical markup	58
6.3	Structural markup (block-level elements)	59
6.3.1	Titles, paragraphs, and quotations	59
6.3.2	Lists	60
6.3.3	Subsections	60
6.3.4	Code listings	60
6.3.5	Figures and tables	62
6.3.6	Admonishments	64
6.3.7	Tasks to do, and remarks when done	64
6.4	Inline markup (element types in mixed content)	65
6.5	Bibliography	70
6.5.1	The <bibliography> element	71
6.5.2	Bibliographic entries	71
6.5.2.1	Title and subtitle	73
6.5.2.2	Authors and editors	73
6.5.2.3	Publication metadata	73
6.5.2.4	Conference metadata	74
7	Producing your class or package	76
7.1	Maintaining your class or package	76
8	References	77
9	The classpack macros and settings	79

9.1	Auto-initialisation	79
9.2	Packages required for the classpack package itself (used <i>only</i> when producing documentation for packages authored using <i>ClassPack</i>)	79
9.3	Index settings	80
9.4	Annotation settings	80
9.5	Table of Contents	80
9.6	Lower-level sectioning	81
9.7	Appendix settings	81
9.8	T _E X and other logos	81
9.9	Formatting additions	83
10	DocBook DTD shim	85
10.1	Set up the invocation and disable HTML forms	85
10.2	Changes to attribute lists	85
10.3	Parameter Entity switch for character entities	87
10.4	Invocation	90
11	The <i>db2dtx.xsl</i> script	91
11.1	XML Declaration, namespaces, and setup	91
11.2	Parameters and variables	92
11.2.1	Versioning	93
11.2.2	Lookup files	93
11.2.3	Naming	94
11.2.4	Document metadata	94
11.2.5	Text control	95
11.2.6	Process control	96
11.3	Creating the .dtx file structure	96
11.3.1	Installer, README, and MANIFEST	97
11.3.2	Copyright	97
11.3.3	Special for class files	98
11.3.4	Special for shell scripts	98
11.3.4.1	Shell call	98
11.3.4.2	Script author	98
11.3.4.3	Script history	99
11.3.4.4	Script version	100
11.3.4.5	Script name	100
11.3.5	File header	100
11.3.5.1	Document type tag	100
11.3.5.2	Constructing the list of packages	102
11.3.5.3	Preloaded packages and options	103
11.3.5.4	Special treatment for packages needing preloading	104
11.3.5.5	Special treatment for conflicting package options	104
11.3.5.6	Language preloads for the babel package	106
11.3.5.7	The ltxdoc \documentclass	108
11.3.6	Automated package management	108
11.3.6.1	Implement the packages	109
11.3.6.2	Load the current package itself if needed	112

11.3.7	Additional setup commands	113
11.3.7.1	From the user document	113
11.3.7.2	Include the bibliography file	113
11.3.7.3	Extras from the prepost.xml file	113
11.3.7.4	DocStrip and L ^A T _E Xdoc settings	114
11.3.7.5	The \documentclass command	114
11.3.7.6	Character-set controls	115
11.3.7.7	Revision history	115
11.3.7.8	Indexing setup	117
11.3.7.9	Deferred commands from the user document	119
11.3.8	Process the document parts	120
11.4	Pattern templates	120
11.4.1	Metadata	120
	Make the title itself	125
	Footnotes for title page	126
11.4.2	Hierarchy	128
	Documentation	129
	Code	130
	Margin width calculation	130
	Code content	131
	Licence	132
	Files	132
	Chapter starts	134
	Chapter ends	135
	Processing	136
11.4.3	Pool	142
	Auto-punctuation for list items	143
	Inline documentation in code	146
	Start the definition	150
	Label the definition	150
	Add to Table of Contents	151
	Additional path conditions	152
	End the definition	153
11.4.4	Lists	162
11.4.5	Flow	170
11.4.6	Special-purpose templates and utilities	211
11.4.7	Text	214
11.4.8	Unhandled element types	215
11.5	Named templates	216
11.5.1	Node-set of elements for <i>AutoPackage</i>	216
11.5.1.1	Conditions for inclusion	216
11.5.1.2	Assemble the elements for each condition	217
11.5.1.3	Create a result tree of packages	221
11.5.1.4	Packages explicitly included by the author	226
11.5.1.5	Constructing the list	227
11.5.2	Other named templates	230
11.5.3	External files	261
11.5.4	Utilities	263

11.5.5	Copyright	276
11.5.6	DocTeX text	279
11.6	Modes	281
11.7	Creating the web page	290
11.8	Ending	297
12	The db2bibtex.xsl script	299
12.1	XML Declaration, namespaces, and setup	299
12.1.1	Citations	299
12.1.2	Bibliographic entries	309
13	The figtab2latex.xsl script	328
14	The db2md.xsl script	355
14.1	XML Declaration, namespaces, and setup	355
14.2	Flow	366
14.3	Typesetting plain text	373
15	The Makefile	379
15.1	Auto-initialisation	379
15.2	Shortcut settings	379
15.3	Document and program dependencies	379
15.4	Script dependencies	380
15.5	Values extracted from the document	381
15.6	Target shortcuts	382
15.7	Transformation to .dtx and creation of the package or class . . .	382
15.8	Creating the documentation	383
15.9	Package the files for CTAN	384
15.10	Package the files as TDS	384
15.11	Chunking and fragmenting	385
15.12	Installations	386
15.12.1	System installation	386
15.12.2	Web installation	386
15.13	Recovery and utilities	386
16	The decommentbb1.awk script	388
17	The chunk.sh shell script	389
17.1	Script code	389
18	The chunk.awk script	393
19	The L^AT_EX Project Public License (v 1.3c)	394
19.1	Preamble	394
19.2	Definitions	395
19.3	Conditions on Distribution and Modification	395
19.4	No Warranty	397
19.5	Maintenance of The Work	398
19.6	Whether and How to Distribute Works under This License	399

19.6.1	Choosing This License or Another License	399
19.6.2	A Recommendation on Modification Without Distribution	399
19.6.3	How to Use This License	400
19.6.4	Derived Works That Are Not Replacements	400
19.6.5	Important Recommendations	400
19.6.5.1	Defining What Constitutes the Work	400
20	Files distributed without annotations	401
20.1	The prepost.xml lookup file	401
20.2	The representation.xml file	401
20.3	The languages.xml lookup file	401
20.4	The readme.xml template file	401
	Change History	402
	Index	407





Note on required and optional features

In this document, the keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL have a specific meaning when shown in THIS TYPESTYLE, and MUST be interpreted as described in RFC 2119 ([Bradner, 1997](#)).

When shown in normal type, these words keep their conventional contextual degree of meaning.

Typographic representation

In this document, the following information items are shown in this way:

Item	Description
class	name of a \LaTeX document class
code	word or phrase in a programming language
\backslash command	name of a \LaTeX 'command' (\TeX macro or control sequence)
environment	name of a \LaTeX environment
filename	name of a file
<i>firstterm</i>	first significant use of a technical term
function	name of a function in a programming language
	a button in a GUI
	a label in a GUI
	a top-level menu entry in a GUI
	a keycap (single key)
template	name of a template in an XSLT script
option	name of an option to a \LaTeX command, environment, class, or package
package	name of a \LaTeX package
$\$$ parameter	name of a parameter to a \LaTeX command or environment, or to an XSLT script
<i>productname</i>	a product name
<u>replaceable</u>	a mnemonic for a user-replaceable string in a paradigm of a command or procedure
systemitem	a computer system item (eg hostname or data value)
\langle tag/ \rangle	a tag name in a markup language (eg XML)
varname	a variable name in a supported language

Latest changes

v.1.28 (2024-02-21)

- Full regression test on all maintained packages. *AutoPackage* methodology being documented separately;
- The code for the widening of the LH margin in mid-document (removed in 1.27) remains in abeyance until some of the discrepancies with fancyhdr headings are resolved;
- The problem with the dox package interpreting `\end{macrocode}` when it occurs in mid-code and mid-line has been temporarily circumvented by splitting the relevant `xsl:text` to output the command in two pieces;
- Additional discrepancy discovered of DocTeX parsing an `\ifcase` command mentioned in a comment behind a double percent signs in the `.dtx` file. The package text has been reworded to avoid this until we find out what's happening.

v.1.27 (2024-01-30)

- Rewrite of the way in which packages are detected and handled for the *AutoPackage* mechanism;
- Removed (commented out) the code for the widening of the LH margin in mid-document to accommodate long macro names. For the moment it uses fixed margins all the way through. The adjustment feature may return if problems with the formatting of other page elements (particularly running headers and footers) can be resolved.
- Ongoing work to find out why the dox package is interpreting `\end{macrocode}` when it occurs in mid-code and *not* at the start of the line.
- Minor corrections to formatting.

v.1.26 (2023-12-30)

- Regression release for revised `db2dtx.xsl` 3.1.

v.1.25 (2023-10-06)

- Added `db:book` alone as an extra test in `attribute-value-match` so that a condition for `book/@status="draft"` would be picked up (all the other tests started too far down the tree).

See p. 402 for earlier changes.

1 Summary of the *ClassPack* code management system

ClassPack is an XML-based class and package management system for \LaTeX described in this document (`classpack.pdf`).

Packages and document classes

For newcomers to \LaTeX , a **document class** is a template (a `.cls` file) — it specifies margins, fonts, spacing, layout, and any extra page and content design features for a particular type of document (eg report, article, book, etc), and any extra commands needed to make use of it; a **package** is a plugin (a `.sty` file) which provides a specific type of formatting (eg a new font or family, an extra feature, a new way to typeset something, a variant on a theme, a utility to simplify complex work, etc) that can be used with many different document classes.

ClassPack itself is neither an actual document class, nor a package for end-users, but an XML system for developers to *generate* \LaTeX classes and packages. It *does* provide a small `classpack.sty` *package* file which provides some presentational styling and utilities used in formatting the documentation like this, generated by *ClassPack*, but it is not used or required anywhere else. For the benefit of readers, this `classpack` package is summarised very briefly in [section 1.2 on the next page](#).

The `classpack` package gets used automatically *only* during the formatting of the *documentation* of *ClassPack*-generated packages and classes. It does *not* (and for safety **MUST NOT**) get used in the \LaTeX code of any packages and classes which you write or maintain in *ClassPack*, only in your *documentation* for them. The users of your classes and packages developed using *ClassPack* do *not* need the `classpack` package at any stage unless they wish to re-typeset your documentation themselves.

1.1 Code management features in *ClassPack*

The features of the development environment are mainly implemented in XML and XSLT, with a few dependencies executed in \LaTeX in the `.dtx` file.

Changes to the *DocBook* DTD: *ClassPack* uses *DocBook*, probably the most popular general-purpose XML markup system for computing documentation. Its widespread use means it is strongly supported by toolchains and has extensive support from vendors. More details are in [section 2.1 on page 13](#)

There are some changes to attribute types (and some new ones added), plus some character entities to facilitate \LaTeX : there are no new element types. You use the attributes to indicate identity, to distinguish between the different parts of your \LaTeX class or package, and to signal required

behaviour. These changes are documented in [section 2.2.2 on page 16](#) and annotated in the code in [section 10 on page 85](#).

Helpful features for authors and maintainers: A number of other features are implemented in the XSLT code rather than in the specifications in the DTD because they require computation more suited to XSLT than \LaTeX (for example, look-ahead). These features are intended to let the author ‘just write’, and have the system handle any underlying complexity. These are documented in [section 6 on page 56](#) and annotated in the code for the scripts in the Appendices.

AutoPackage: This is an automated mechanism added in 2013 for detecting and including required \LaTeX packages in documentation, and for normalising the inclusion of packages in your own classes and packages. It lets you create your own preferred way of working for how you write documentation and design your classes and packages. This is documented in [section 5.4 on page 52](#) and annotated in [section 11.5.1 on page 216](#).

Packaging: The standard *make* (1) Unix/GNU Linux utility (Makefile) included with *ClassPack* builds the .dtx and .ins files, typesets the PDF documentation, installs the class or package in the author’s Personal \TeX Directory, and creates both CTAN and TDS zip files for distribution. Details are annotated in [section 15 on page 379](#). There is currently no equivalent method implemented for Windows.

1.2 Presentation features in the classpack package itself

The formatting of *ClassPack* documentation implements the following differences from standard Doc \TeX :

1.2.1 Features adjusted for ease of reading

Details of the annotated code are in [section 9 on page 79](#). They cover the following formatting changes:

1. Two-column index instead of three-column;
2. Dark Blue colour for annotated code;
3. Recalculation of the left-hand margin in documentation to accommodate long variable names;
4. Wider space for the section numbers and page numbers in the Table of Contents, and ragged-right setting for section titles to prevent hyphenation;
5. Appendixes (used for annotated code for ancillary files) are styled at section level, not chapter level;
6. Definitions for the Con \TeX t, X \TeX and X \LaTeX , and Lua \TeX and Lua \LaTeX logos (borrowed from the TUGboat style);
7. Fake small caps (also from TUGboat) for the Bib \TeX logo for fonts without them;

8. New struts for adjusting table spacing;
9. Some hyphenation oddities fixed (see code).

1.2.2 Features adjusted for ease of writing

1. A fix for \LaTeX 's broken description environment, using enumitem;
2. The array package for re-spacing tables;
3. Menu items in documentation now use the graphical features of the menukeys package;
4. Raggedright setting of footnotes from the footmisc package;
5. Running headers and footers from the fancyhdr package;
6. The `\RaggedRight` (hyphenating) ragged-right setting for bibliographies from the ragged2e package;
7. The graphicx package is added to allow the definitions for X_{TeX} etc to work;
8. A counter to be used when calculating the number of items in a list.

An end-user of your classes or packages SHOULD NOT normally need the classpack package unless they want to re-typeset the documentation.

2 Introduction

\LaTeX document classes (layout templates) and packages (formatting styles) are traditionally distributed as pairs of `.dtx` and `.ins` files, written to the specifications and recommendations of the `doc`, `ltxdoc`, and `clsguide` packages.

- The `.dtx` (Doc \TeX) file is a *literate-programming* document, containing modular code and documentation interleaved in such a way that each fragment of code and its explanation are adjacent.

This broadly parallels the *web* literate-programming system due to Knuth (1984) used in the production of the original \TeX system and its documentation;

- The `.ins` file is an installer: when run through \LaTeX , it extracts the programming code from the `.dtx` file into the relevant class (`.cls`), package (`.sty`), and other files;
- Running \LaTeX on the `.dtx` file itself extracts and typesets the documentation.

The construction of a `.dtx` document is quite complex, with a special set of tags and conventions to allow documentation to be separately identifiable to code. The file format relies on the documentation being shielded by a leading percent-space armour (`%_`) on each line to prevent it being interpreted as part of the code; and the environment surrounding the code itself must be shielded by four such spaces (`%_ _ _ _`). Apart from the documentation, the treatment of the code and control statements resembles more a data specification (which in some ways it is) than a conventional text document.

2.1 XML

XML, particularly in its traditional ‘document’ mode, as distinct from its use as an exchange format for rectangular data, offers many similar features to \LaTeX (for example, the named identification of document components), but with a rigid and invariable syntax that can be checked programmatically by any validating XML processor. By contrast, a \LaTeX document (and more specifically, a `.dtx` document) can only be proved by running it through \LaTeX itself: there is no equivalent to the ‘pre-flight’ type of standalone parsing or validating available with XML, although there are several *lint*-like utilities which check basic syntax (Even, 2002; Grothmann, 1993; Thorup et al., 1997), and some \LaTeX editors include syntax-checking.

The *DocBook* vocabulary of XML is designed for technical documentation in computing. It provides markup both for textual material and for data-like structures that occur in computer documentation, making it a suitable candidate for describing a literate-programming type of document such as Doc \TeX .

2.2 ClassPack

The *ClassPack* system is an ongoing experiment or work-in-progress using *DocBook* XML as the storage format for \LaTeX class and package source code and

documentation, and the XSLT2 language to transform the XML file into the pairs of .dtx and .ins files traditionally used for L^AT_EX class and package installation. There are several advantages to this approach:

- XML’s syntax and document construction is extremely robust, and the design of the language means that an XML file can be machine-checked for errors of syntax and construction at the editing stage;
- XML markup is traditionally self-descriptive, with element types being named according to what they are intended to contain.¹ For example, in *DocBook*, a variable name (for example a L^AT_EX length like `\pagelen`) can be marked up as `<varname role="length">pagelen</varname>`, and DocT_EX can flag it (unheralded, ie no backslash, dollar sign, etc) in the margin.

pagelen

(While it is perfectly possible to create a `\varname` control sequence (macro) to do the same in L^AT_EX, it is rarely done by authors. Instead, authors have typically preferred to use visual formatting like `\textit{foo}` for italics or `\verb+foo+` for monospace type. This method means the variable reference is not immediately identifiable programmatically as containing a variable name — it could be anything.);

- Given suitably-descriptive markup and a modular construction, sharing document fragments between applications can be done programmatically. Thus a fragment implementing a L^AT_EX feature (with its associated documentation) can be re-used in other class and package applications *at the XML level* (eg with XInclude, or as an external parsed entity) without the need for manual cutting and pasting, and with the assurance that the latest version is always being included;
- There is a very wide range of software (editors and processors, both free and non-free) available to handle XML documents, including a lot of useful tools for validation, document management, and information extraction.

Everything comes at a cost. The drawbacks of using XML for this include:

- It’s another language to learn. Despite being so widespread, XML and XSLT are not yet common skills in either the programmer community or the L^AT_EX community of package authors or maintainers;
- Programmers in particular believe they dislike [‘document’] XML because it’s a markup language, not a programming language (although xsl is implemented in XML), and the syntax is different from that of most programming languages.

(There is a cure for this: ask them to write their thesis in JSON and XML and judge which one is more suitable.);

- Although there is plenty of software for editing XML, it is not well-developed for writing or editing text documents in synchronous typographic form (often misleadingly called WYSIWYG) (Flynn, 2014); despite many recent advances,

¹This approach has long been a source of philosophical contention, however; see Holman (2013) and Kay (2013).

even the best or most expensive editors are designed for XML experts, not for the average user (Flynn, 2009) and not even for the sophisticated user like a \LaTeX class or package author or maintainer.

The *ClassPack* framework is enabled by three main software components:

1. An XML vocabulary (a DTD) used for naming the component parts of documentation and code, and specifying where they belong and how they fit together. This is implemented in the file `doctexbook.dtd`, which is a lightly-modified *DocBook5* documented in [section 2.2.1](#) and [section 2.2.2 on the next page](#);
2. An XSLT script to implement the logic of combination and separation needed to create the `.dtx` and `.ins` files. This is implemented in the files `db2dtx.xsl`, `db2bibtex.xsl`, `figtab2latex.xsl`, and `db2md.xsl`, annotated in the Appendices and explained throughout this document;
3. A lookup table of commonly-used \LaTeX packages, with brief descriptions, setup commands, and an optional set of dependency specifications to automate their invocation with the *AutoPackage* feature. This in essence a database implemented as the XML file `prepost.xml`, and documented in [section 5 on page 48](#).

This file can be edited by individual class/package authors or maintainers to reflect their personal tastes and way of working.

2.2.1 Use of XML

The XML vocabulary used is *DocBook* (Walsh, 2011), which is in widespread use for the documentation of computer systems, and is well-supported on all platforms. The current *ClassPack* system uses version 5. Although it is highly modular and easily adapted for many purposes, only a few minor changes have been made for its use, but a number of element types have been put to uses not envisaged by the developers. The current version implements only a DTD; a future version will also use RNG, and will create a customisation layer to replace the current DTD shim.

This habit of using (some say, abusing) XML markup for different purposes than intended is very common, and widely deprecated because it is usually undocumented and because the resulting schema is arguably not the type of document the original designer[s] envisaged. This document explains in [section 10 on page 85](#) what has been [ab]used and for what reasons. Once *ClassPack* settles down, a more formal expression of the vocabulary can be made from the RNG source, removing the parts that are not required, and making it simpler to edit with.

The adaptation of *DocBook* in this version subsists mainly in the addition of some attributes and entity declarations to allow the conversion to \LaTeX of special characters and other features that would otherwise involve extensive re-parsing of the character data content (text). The changes also implement a few modifications to the way the \LaTeX code is output by the XSLT script for typographical purposes.

The modifications to the DTD explained below are annotated in [section 10 on](#)

page 85, and the file is extracted as `doctexbook.dtd` when processing this document class, and it can be referenced as such in your *ClassPack* documents (see [section 3.1 on page 19](#)).

2.2.2 Changes to the DocBook DTD

The changes to the *DocBook5 DTD* are mainly to enable the reinterpretation of some element types and attributes for use in the DocTeX environment. They also add some aspects of linking, cross-referencing, annotating, and formatting that are not normally provided for in *DocBook*.

Some of the changes should be considered ‘tag-abuse’, as they hijack existing attributes for unconventional purposes. A separate and more fundamental set of changes is planned for a future date, using a shim for RNG.

1. `<html:form>` is not used;
2. The `<date>`, `<year>`, `<confdates>`, and `<biblioid>` element types get a mandatory `@YYYY-MM-DD` attribute (suffixed `-from` and `-to` in the case of conference dates), so the elements SHOULD BE used EMPTY (content will be ignored). The attribute value MUST be a valid ISO 8601 date. For `<biblioid>` this is the date the resource was retrieved;
3. The `<blockquote>` and `<quote>` element types get optional `@units`, `@begin`, and `@end` attributes for specifying page or other ranges when a citation key is provided in `@linkend`, eg `units="p" begin="22" end="34"`;
4. The `<xref>` and `<biblioref>` element types’ attribute `@linkend` is changed from `<IDREF>` to `<IDREFS>` to allow multiple citations at a single point;
5. The `<bibliography>` element type is given a `@label` and `@xreflabel` for the name of the reference style and any ancillary style file name, respectively.
6. On the `<biblioentry>` element type, an `@xml:id` attribute is mandatory, and the `@xreflabel` attribute MUST contain one of the valid biblatex entry types;
7. For code annotations, the `<annotation>` element type has a mandatory `@role` attribute to provide the type of object being documented. See the list in [section 3.6.1 on page 28](#) for a list of values. The name of the object being documented MUST be given in the `@xreflabel` attribute, and a new attribute `@reuse` is added to provide the `<IDREFS>` of annotations to copy from elsewhere (see [section 3.6.1.2 on page 29](#));
8. The `<programlisting>` element type now has an `@endinglinenumber` attribute for reference to segments of an external file being annotated;
9. The `<remark>` element type (used for flagging changes) has mandatory `@version` and `@revision` attributes which MUST match a version number and a revision date in the revision history;
10. A list of computer languages supported by the listings package (as extended in the `prepost.xml` database) is defined, and applied as a `@language` attribute

on `<classname>`, `<cmdsynopsis>`, `<command>`, `<envvar>`, `<function>`, `<literal>`, `<literallayout>`, `<option>`, `<programlisting>`, `<screen>`, `<varname>`, and `<parameter>` element types. The default value is LaTeX (warning: not LaTeX);

11. The `<varname>` element type gets a `@role` attribute to hold the type of variable being documented (`counter`, `length`, `prog`, `color`, `template`, `variable`, `parameter`, `field`, and `entrytype`);
12. The `<type>` element type is hijacked wholesale to handle typographic variance for use in annotating typographic changes. Eight attributes are added:
 - (a) `@fontencoding` (T1, OT1, U) as the only ones needing to change from the default;
 - (b) `@fontface` is *either* a ‘Karl Berry’ typeface name (see the `fontname` package for details) *or* the name of a typeface to be used locally;
 - (c) `@fontfamily` is one of `roman`, `sans`, or `monospace`;
 - (d) `@fontseries` is one of `k`, `l`, `r`, `m`, `n`, `b`, `bx`, `it`, `sl`, `ll`, `c` as defined in the `fontname` package;
 - (e) `@fontshape` is one of `up`, `it`, or `sc`;
 - (f) `@fontsize` is a size in a dimension known to LaTeX;
 - (g) `@fontalign` is `yes` or `no` (default is `no`, currently unused);
 - (h) `@fontcolor` is a color name known to the `xcolor` package or defined earlier in the document.
13. The `<seglistitem>` has an `@omit` attribute to specify temporary omission.

A Parameter Entity switch is used to list both LaTeX and HTML expansions of a long list of character entities.

3 Setting up a *ClassPack* document

Using *ClassPack* to create and maintain \LaTeX classes and packages requires the following initial steps. These are only done once, at the start of a new class or package.

In the current version it is **ESSENTIAL** that you keep each class or package you manage in a separate directory, named after the class or package. Specifically, the directory name, the XML filename, and the `@xml:id` value of the root element **MUST** all be the same name.

A file `template.xml` is provided to help you get started: it contains comments and ‘placeholder’ values to help the construction of a new document. Create a new directory for the class or package, open `template.xml` in your XML editor, and save it with the same name and the filetype `.xml` in the new directory.

Tooling up

You need the following tools:

- a full installation of \LaTeX ;
- this package;
- an XML editor: I use *Emacs* with *psgml-mode* and *xxml-mode*, but any competent XML editor will do;
- a copy of the *DocBook5* DTD from <https://docbook.org/xml/5.0/dtd/>;
- an XSLT processor: I use *Saxon* (which in turn requires *Java*);
- a PDF reader;
- some XML tools: I find the *LTxm/2* toolkit from Edinburgh University invaluable for ad-hoc querying of documents. It also gets used in the *Makefile*.

To start a new document, you **MUST** set up:

1. the configuration (see [section 3.3 on page 21](#)) to specify the name, date, version, type, audience, status, and other key values in the root element;
2. the documentary metadata (see [section 3.4 on page 24](#)) such as the package title, author name[s], contact details, abstract, and the initial entry in the revision history, exactly as you would do for an article or any document (including a `.dtx` file) that you would edit manually;
3. two *separate* lists of packages:
 - (a) any packages you need for the documentation that will *not* be invoked automatically by *AutoPackage*;
 - (b) any packages needed by the class or package code itself.
 (see [section 4 on page 39](#)). It is *essential* that these are set up exactly.

Note that you do *not* put the list of packages required by the class or package itself (item 3b in the list on p. 18) as manual `\RequirePackage` commands in the annotated code of your class or package. You give them separately, and specify where in your code they are to be included, as described in [section 4 on page 39](#).

You MAY also provide any extra L^AT_EX commands you need for your documentation (see [section 4.4 on page 45](#)), although the introduction of the `prepost.xml` *AutoPackage* database or lookup file lets you automate much of this.

When you update a class or package already in *ClassPack*, you need to change:

1. the version number and/or revision number in the root element attributes `@version` and `@revision` (see '`@version`' on page 22);
2. the corresponding `@version` attribute on the relevant `<revision>` element in the revision history to match the version and revision above (see '`<revhistory>`' on page 26);
3. the DocT_EX checksum in the root element `@security` attribute, if used (see '`@security`' on page 23).

If these values do not all match, the class or package will not build.

3.1 XML editing

To create or edit a *ClassPack* XML document you MUST have a copy of the *DocBook5* Document Type Definition (DTD) and the *ClassPack* DTD customisation shim (`doctexbook.dtd`) installed on your computer in a location that is known to or visible to your XML editor. See your editor's documentation for how to do this — the `template.xml` file assumes that the `doctexbook.dtd` file is in the directory (folder) *above* the document you are editing (see the note in [section 3 on the previous page](#)).

1. The *DocBook5* DTD can be downloaded from the [DocBook web site](#) ;
2. The file `doctexbook.dtd` is included with this classpack package.

In extreme cases putting the DTDs in the same directory as your *ClassPack* document will work if you edit its location in the Document Type Declaration (the line beginning `<!DOCTYPE book`), but it means you would need a separate copy in every class or package directory you maintain.

The Document Type Declaration illustrated below is for reference only: the included file `template.xml` already has this set up, but you MUST edit the location of the DTD shim `doctexbook.dtd` to be wherever you keep it on your system, and edit that file to specify whereabouts the main `docbook.dtd` file is on your system.

Your class or package documents SHOULD start with the XML Declaration giving the version of XML as `"1.0"`. With a DTD, you MUST give the **Document Type Declaration** on the next line of your XML file. If your XML editor omits the XML Declaration, then your Document Type Declaration will of course be the first line

of the file. There are two forms for expressing the Document Type Declaration (different editors may use one or the other):

either: with the PUBLIC keyword and the Formal Public Identifier (FPI) shown, plus the system filename of the DTD:

```
<?xml version="1.0"?>
<!DOCTYPE book PUBLIC
    "-//Silmaril//DTD DocBook 5.0 for DocTeX ClassPack//EN"
    "../doctexbook.dtd">
```

or: you can omit the FPI and specify the DTD after the SYSTEM keyword:

```
<?xml version="1.0"?>
<!DOCTYPE book SYSTEM "../doctexbook.dtd">
```

The DTD file `doctexbook.dtd` can be anywhere on your system, provided your editor is capable of reading it: in these examples it is assumed to be in the directory above the one where your document is. If you store it elsewhere, just give the full filepath, for example:

```
<?xml version="1.0"?>
<!DOCTYPE book SYSTEM "/usr/local/lib/xml/dtds/doctexbook.dtd">
```

You will also need to edit the location of the *DocBook* DTD in your copy of `doctexbook.dtd` because it will be in a different place on your system. The distribution file says:

```
<!ENTITY % db5dtd SYSTEM
    "/dtds/docbook/docbook-5.0/dtd/docbook.dtd">
```

Just change this to wherever your copy is stored.

Now when you open your document, any conformant XML editor will find the `doctexbook.dtd` and read it, so that it then knows the names of all the element types that you can use, and how they fit together into a document, and will create menus for adding markup as you write.

3.2 *ClassPack* document structure

Every XML document **MUST** have one outermost enclosing element (the ‘root’ element) which contains everything else. The root element type used in a *ClassPack* document is `<book>`, one of the built-in document types provided by *DocBook*.

Within the `<book>` element there are five main areas:

configuration: attributes on the `<book>` element itself which hold the configuration data for your class or package; see [section 3.3 on the next page](#).

metadata: (information *about* the document) is held in an `<info>` subelement; see [section 3.4 on page 24](#);

user documentation: is held in a `<part>` element with the `@xml:id` attribute "doc"; see [section 3.5 on page 26](#);

annotated code: is held in a second `<part>` element, with the `@xml:id` attribute "code"; see [section 3.6 on page 27](#);

unannotated ancillary files: OPTIONAL; are held in a third `<part>` element with the `@xml:id` attribute "files". These are files that are to be recreated as-is when the package or class is installed by a user, such as sample data or example documents (see [section 3.7 on page 30](#)).

The two `<part>` elements with the specific ID values "doc" and "code" are COMPULSORY and MUST occur exactly once each only.

```
<?xml version="1.0"?>
<!DOCTYPE book SYSTEM "doctexbook.dtd">
<book xml:id="...">
  <info coverart="">...</info>
  <part xml:id="doc">...</part>
  <part xml:id="code">...</part>
  <part xml:id="files">...</part>
</book>
```

Within each `<part>`, there is a specific structure used:

doc: can use prefaces, chapters, sections, subsections, tables, figures, paragraphs, lists, block quotes, cross-references, bibliographic references, etc as any normal text document does;

code: your principal class or package goes in one or more `<chapter>` elements; any additional *annotated* files each go in separate `<appendix>` elements;

files: each *unannotated* file goes in its own `<chapter>` element.

3.3 Configuration for a new class or package

The `<book>` element is the outermost container for the document. Its attributes are used to carry the configuration information, for example:

```
<book xml:id="classpack" version="1" xml:base="classpack"
  remap="12pt" arch="package" audience="lppl" condition="2015-01-01"
  conformance="LaTeX2e" os="all" revision="10" security="259"
  userlevel="sty" vendor="Silmaril Consultants"
  annotations="\raggedright" xlink:role="">
```

@xml:id: This MUST be the one-word name of the class or package. It will be used as the \LaTeX filename (with `.cls` or `.sty` and `.dtx` and `.ins` added automatically), so it SHOULD by preference be all lowercase and MUST consist of letters, digits, and hyphens only, starting with a letter. Spaces and other types of character are not allowed here.

The XML rules for IDs require this restriction, which means *ClassPack* cannot be used to maintain a class or packages whose name begins with a digit.

@arch: The ‘architecture’ of the document, which defines the type of file you are going to produce; this **MUST** be either “class”, “package” or “script”. This **MUST** correspond with the value of the <user level> attribute below, which provides the filetype.

@user level: The filetype of the output class or package document or script; this **MUST** correspond with the value of the <arch> attribute above, and be either “cls”, “sty”, or a script language extension such as “bash” (the Bourne-Again Shell, the only value supported in this version), “py” (*Python*), “pl” (*Perl*), etc.

@version: The major version of the class, package, or script. Conventionally, early-development (α) or pre-release (β) versions of software start at version zero. See <status> below;

@revision: The sub-version or release of your class or package or script. This is combined with the major version number above, separated by a dot, to produce the complete version number.

The most recent revision history entry gets tested against this when the file is processed, and an error message is displayed if the version numbers do not match, as a warning that you have updated one without updating the other, and may therefore have forgotten to document a change (see the <revision> element in ‘<revhistory>’, the last item in the list in [section 3.4 on page 26](#)).

@status: The development status of the class or package, eg “alpha”, “beta”, “rc” (release candidate), “draft”, “final”, etc. Use the letter ‘p’ for production versions.

@conformance: The T_EX format required to process the .dtx document. Only the value “LaTeX2e” is supported at the moment.

@condition: The version of the format identified in @conformance, expressed as an ISO 8601 date (yyyy-mm-dd).

Note that this is *not* in the L^AT_EX format (yyyy/mm/dd).

@os: The operating system[s] for which the class or package is relevant. Currently, only the value “all” is supported.

@audience: The licence under which the class or package is made available. For normal publicly-available L^AT_EX classes or packages which will be uploaded to the Comprehensive T_EX Archive Network (CTAN), use the value “lpp1” (L^AT_EX Project Public Licence).

A copy of the LPPL (currently version 1.3c) is distributed with *ClassPack* in a file called lpp1.xml, which must be copied or soft-linked (aliased) to each directory in which you process *ClassPack* documents needing to use it.

Classes or packages for private or commercial use will probably need to use another value, but the value is used as a filename, so a .xml file with that name **MUST** exist in the directory in which your *ClassPack* document is processed. It **MUST** contain a *DocBook* <chapter> element containing the text

of the licence, which will be included at the end of the documentation. See the supplied `lpp1.xml` as an example.

@security: The checksum value emitted by `ltxdoc` when it processes your class or package to format your documentation. See the documentation for the `ltxdoc` package for details.

Setting this to zero avoids the \LaTeX error message during early development, when every edit would change the checksum. As with the version values, you **MUST** update this value, if non-zero, to match the one that `ltxdoc` reports.

@vendor: Your name, or the name of the organisation responsible for the work on this class or package.

@remap: Any class options to pass to the `ltxdoc` package, such as `a4paper`, `12pt`, etc.

This, and the following `<annotations>` attribute, make it easier to change the global formatting of the documentation.

@annotations: Any \LaTeX commands required for global application at the start of the documentation that cannot easily be included anywhere else (eg `\raggedright`, `\sffamily`, etc);

@xml:base: The name of the *subdirectory* where the resulting `.cls` or `.sty` file should be installed in a TDS-compliant \TeX installation, *relative to the `texmf/tex/latex` directory*. For most packages, this means the name of the directory you want created by the installation `.tds.zip` file, in which the `.cls` or `.sty` file will be put. This **SHOULD** normally be the same as the name of the class or package, and therefore the same as the name of the ID value in the `@xml:id` attribute above.

@xlink:role: OPTIONAL. If this is present (null or non-null), *and* the document being written is a package, not a class, then the package itself will be included in the \LaTeX Preamble for the documentation (usually so that it can be used in examples). If a value is given, it will be used as an optional argument to the generated `\usepackage` command.

If the package is required with no options, use this attribute but set it to null (eg `""`). The date constraint is added automatically, using the current package date as defined by the most recent entry in the `<revhistory>` (see '`<revhistory>`', the last item in the list in [section 3.4 on page 26](#)).

Note that this only works for packages, not classes. Classes cannot not be documented using themselves in *ClassPack*.

It is essential to get these values correct, otherwise subsequent processing will produce unexpected results, or no results at all.

Note that the experimental use of `@xreflabel` in earlier versions to define the \LaTeX processor to be used for the documentation has been moved to the `@conformance` attribute on the `<part>` start-tag for documentation (see [section 6.2 on page 58](#)).

3.4 Metadata

The titling, specification of packages (separately for the documentation and for the class or package itself), revision history, abstract, copyright, and availability are all kept at the top of the document in an element called `<info>`, immediately inside the `<book>` start-tag:

```
<info coverart="somepic.jpg">
  <cover>...</cover>
  <!-- THE METADATA STARTS HERE -->
  <title>XML mastering for &LaTeX; document classes...
  <author>...
  <copyright>...
  <releaseinfo>http://latex.silmaril.ie/packages/</releaseinfo>
  <annotation>...
  <abstract>...
  <revhistory>...
</info>
```

Of these, the `<cover>` element is the most complex, as it is used to hold all the details of packages and settings required for the class or package *and* for the production of the documentation. It therefore gets the whole of [section 4 on page 39](#) to itself.

The other metadata element types within the `<cover>` element are more obvious, and use the conventional *DocBook* structure.

<title>: This contains the title of your class or package in natural language.

This is *not* the content of the \LaTeX `\title` command in the `.dtx` file, which is automatically preset to the standard phrase ‘The name \LaTeX 2_ε document class’ (or ‘package’) where name is the name of your class or package as specified in the `@xml:id` attribute of the `<book>` root element.

The title you give in *this* `<title>` element is the explanatory *subtitle* which appears *under* the automatically-generated one on the first page of the typeset documentation.

<author>: The `<author>` element provides identity markup for the author[s], using a `<personname>` element for each author, containing subelements for `firstname`, `surname`, and other forms of naming; optionally followed by an `<affiliation>` element, where you can identify employer or other status; address; email; and URI, as in the following example. The name, affiliation, and email address are used in the `\author` command of the `.dtx` file.

```
<author role="maintainer">
  <personname>
    <firstname>Peter</firstname>
    <surname>Flynn</surname>
  </personname>
  <affiliation>
    <orgname>Silmaril Consultants</orgname>
    <orgdiv>Textual Therapy Division</orgdiv>
  </affiliation>
  <address>Cork, Ireland</address>
```

```
<email>peter@silmaril.ie</email>
<uri>http://blogs.silmaril.ie/peter</uri>
<contrib role="sponsor">UCC</contrib>
</author>
```

For multiple authors, you must enclose multiple `<author>` elements (one per author) in an `<authorgroup>` container element, for example:

```
<authorgroup>
  <author role="maintainer">
    ...
  </author>
  <author>
    ...
  </author>
  ...etc...
</authorgroup>
```

At least one of the authors must be identified as the maintainer of the package or class, by adding the `<role>` attribute with the value "maintainer".

A `<contrib>` element with a `@role` attribute set to "sponsor" can be added in an `<author>` block, to give the name of an organisation sponsoring the development of the class or package.

<copyright>: This element lets you identify the `<year>` and the name of the `<holder>` (you, your employer, or some other entity).

```
<copyright>
  <year YYYY-MM-DD="2020"/>
  <holder>ACME Consulting</holder>
</copyright>
```

<releaseinfo>: In the absence of other ways of identifying where to find your class or package (assuming it will eventually find its way onto CTAN), this element can be used to hold the URI of a location where it can be downloaded, such as your personal or business web site.

<annotation>: This element is used for a warning or notice you want placed in the Preamble of the `.ins` file, which is used for extracting your class or package from the `.dtx` file, where it will be seen by users installing the software.

This use case of `<annotation>` in this location is *different* from the one described in [section 3.6.1 on page 27](#).

<abstract>: The Abstract is formatted on the front page of your documentation. Like any abstract, it should summarise what the class or package does, and who might want to use it.

The `<abstract>` element MAY start with an optional `<title>` element, which (if present) will be used to change the value of the `\abstractname` in the .dtx file ('Summary' is a common choice).

The rest of the abstract is basically just paragraphs; note that lists, block quotations, figures, tables, etc cannot be used in a *DocBook* Abstract.

<revhistory>: This holds the top-level information about each major and minor revision, outlining the main changes. The version number of the most recent revision, as identified by the latest value of the `@YYYY-MM-DD` attribute of the `<date>` element, MUST match the version number composed from the major version and the revision in the `<book>` root element attributes (see '**@revision**', the fifth item in the list in [section 3.3 on page 22](#)).

```
<revision version="0.72">
  <date YYYY-MM-DD="2012-02-11"/>
  <revdescription>
    <itemizedlist>
      <title>Wrote internal documentation</title>
      <listitem>
        <para>Created the classpack.xml template
          as an example.</para>
      </listitem>
    </itemizedlist>
  </revdescription>
</revision>
```

The four most recent entries are printed after the Table of Contents as 'Recent Changes'.

Comments on individual changes to the code should be documented *at the code location*, using the `<remark>` element (see [section 7.1 on page 76](#)), eg

```
<remark version="0.70" revision="2018-05-29">Added timestamp</remark>
```

These `<remark>` elements get collated as `\changes` commands *in your class or package*, and gathered together in the changelog by ltxdoc during processing, along with all the other revision elements.

3.5 Part "doc": the user documentation

The documentation for the end user is normal descriptive text which explains how to use the package or class. It is, in effect, the manual for your class or package, aimed at the L^AT_EX user (author) who needs to know what it does and how it works.

3.5.1 Hierarchical structure

Your documentation goes in the *ClassPack* "doc" part in `<chapter>` elements, which can contain `<sect1>` sections, which in turn can contain `<sect2>` subsections, and so on. This hierarchical structure is described in [section 6.2 on page 58](#).

Because the latexdoc package is based on the article package, the *ClassPack* <chapter>s actually become latexdoc \sections and the <sect1>s become latexdoc \subsections, etc, for the purposes of formatting the documentation.

3.5.2 Structural markup

Within your chapters and sections, you can use most of the conventional structural features of any DocBook document: paragraphs, lists, figures, tables, and examples of code. Note that code examples in the documentation part of a *ClassPack* document do *not* get extracted into the class or package file.

Most of the specialist constructs of *DocBook* for documenting programming languages are *not* used in *ClassPack* except for a few which are hijacked in the <info> section for setup purposes, and some [ab]used in the prepost.xml database file for technical reasons. The pool of structural markup elements supported is described in [section 6.3 on page 59](#).

3.5.3 Inline markup

Within the text itself (eg in paragraphs) there are many elements supported to allow you to identify and format your documentation with minimal effort. These inline or ‘flow’ elements are described in [section 6.4 on page 65](#). Many of them implement time-saving automation features.

3.6 Part “code”: the annotated code

In the code <part>, you provide the actual L^AT_EX code of your class or package, separated into chunks that you can annotate to explain how it works or why you wrote it that way. You can use <chapter>s and <sect1>s etc to structure your annotations in a readable manner: this has no effect on the class or package code extracted into the .dtx file, only on the formatting of the printed annotation.

3.6.1 Annotations

Your code SHOULD be documented in a modular way using the <annotation> element to identify the macro or environment or other object you are describing, using the <role> and <xreflabel> attributes. You then give the explanatory annotation in a <para> element and the actual code in a <programlisting> element. Here is an example of the documentation of a rewriting of the macro \MacroFont (the output is given on the code on the following page and on [page 29](#)):

```
<annotation role="macro" xreflabel="MacroFont">
  <para>The <package>doc</package> and <package>docx</package>
    packages use the <command>MacroFont</command> command for
    the marginal labelling of code annotation. We add here the
    colour DarkBlue from the <option>svgnames</option> option
    to the <package>xcolor</package> package.</para>
  <programlisting>
\def\MacroFont{\fontencoding\encodingdefault
  \ttfamily\fontseries{m}\fontshape\updefault
```

```
\small\selectfont\color{DarkBlue}}%
</programlisting>
</annotation>
```

Each such `<annotation>` element SHOULD describe a single macro or environment or other code object such as a dimension, counter, etc listed below. In the default DocTeX system, only macros and environments can be identified (the macro and environment values, see Mittelbach (2016, p 6)).

macro: the original DocTeX usage;

environment: the original DocTeX usage.

ClassPack extends this to handle the following list of values for the `@role` attribute of the `<annotation>` element for L^AT_EX and other related structures. These are the values declared in `prepost.xml` as extensions to the doc package.

attribute: describing an attribute in an XML document;

attvalue: describing an attribute *value* in an XML document;

box: describing a new box;

class: describing a new class;

color: defining a new colour;

comment: describing a comment inserted in a file;

counter: describing a new counter;

dtd: documenting a DTD;

element: documenting an XML element type declaration;

entity: documenting an XML entity declaration;

error: defining an error;

field: documenting a data field (eg in BibTeX);

file: describing a new file;

font: describing a new font (not a new typeface; see below);

function: documenting a function in a language;

language: describing a new [computer] language (eg in listings);

length: describing a new length;

mode: documenting a mode of operation (eg in XML);

option: defining a new option;

package: defining a new package;

parameter: describing a parameter passed to a module (eg in XSLT);

rubric: defining a new rubric;

setting: describing a setting or block of settings such as initial values or switches ;

switch: describing a new switch (TeX’s `\newif`);

template: describing a template (eg in XSLT);

typeface: describing a new typeface;

variable: describing a variable in a language.

In the above example, the result of processing the XML *ClassPack* document is a fragment in the `.dtx` file:

```
% \begin{CPK@macro}{\MacroFont}\label{ann-MacroFont}
% The \textsf{doc} and \textsf{docx}
% packages use the {\ttfamily}\textbackslash{}MacroFont} command for
% the marginal labelling of code annotation. We add here the
```

```
% colour DarkBlue (from the \textbf{\texttt{svgnames}} option
% to the \textsf{xcolor} package).\par
% \begin{macrocode}
\def\MacroFont{\fontencoding\encodingdefault
\ttfamily\fontseries{m}\fontshape\updefault
\small\selectfont\color{DarkBlue}}%
% \end{macrocode}
% \end{CPK@macro}
```

The result of extracting the .sty file during installation is therefore the \LaTeX code...

```
\def\MacroFont{\fontencoding\encodingdefault
\ttfamily\fontseries{m}\fontshape\updefault
\small\selectfont\color{DarkBlue}}
```

...and the result of typesetting the .dtx file is, of course, the typeset text in the documentation:

The doc and docx packages use the `\MacroFont` command for the marginal labelling of code annotation. We add here the colour `DarkBlue` from the `svgnames` option to the `xcolor` package.

There are two special cases of the `<annotation>` element: colours and reusability:

3.6.1.1 Specifying colours : When using an `<annotation>` element to document a colour (eg a `\definecolor` command from the `xcolor` package), in addition to the `@role` and `@xreflabel` attributes described above, you MUST add the the colour space (eg `rgb`, `cmyk`, `HTML`, etc, as defined in the `xcolor` documentation) in the `@arch` attribute; *and* the colour code to use in the `@audience` attribute. This is required to allow DocTeX to predefine the colour for use in the documentation itself, as well as having it defined for your class or package. For example (from the euflag package document)

```
<annotation role="color" xreflabel="PantoneReflexBlue"
arch="HTML" audience="003399">
<para>As specified in the web page.</para>
<programlisting>
\definecolor{PantoneReflexBlue}{HTML}{003399}
%\definecolor{PantoneReflexBlue}{cmyk}{1.00,.67,0,.40}
</programlisting>
</annotation>
```

3.6.1.2 Reusability : To avoid the need for annotations to be repeated verbatim in annotated files such as .sty files that accompany a .cls file, there is a `@reuse` attribute on the annotation element type, declared as `<IDREFS>`. This is for a space-separated list of `@xml:id` values of *existing* annotation elements elsewhere in the *same* document. The relevant annotations will then be included in at the new location as well, exactly as if they had been physically present in the source code. When using this attribute, the `<annotation/>` element MUST be EMPTY.

Work is ongoing to support the use of XLink to allow this type of retrieval across different documents.

3.6.2 Ancillary annotated files

Ancillary files which you want to annotate *and* have extracted along with the class or package file **MUST** each go in their own `<appendix>` element, starting after the last `<chapter>` element of the package or class in the "code" part. You write them and code them exactly as for the main code of your primary file in the preceding chapter[s]. Each such `<appendix>` element **MUST** contain at least one `<sect1>` element where you put the documentary `<annotation>`s.

These files would typically be style packages, scripts, or data samples accompanying a class or style file, whose code needs documenting. Files for extraction which do *not* need documenting are called *unannotated* ancillary files. They **MUST** be stored in the "files" `<part>` as described in [section 3.7](#).

The `<appendix>` element **REQUIRES** the following attributes:

@xml:id: This is a unique identifier for the file. It **MUST NOT** be the filename.

@arch: This describes the architecture of the file. The range of values is as given in '@arch', the second item in the list in [section 3.3 on page 22](#) plus the value "document".

@userlevel: This describes the type of file being extracted. The range of values is as given in '@userlevel', the third item in the list in [section 3.3 on page 22](#) plus values such as "dtd", "xsl", or "awk", corresponding to the output filetype. See also below.

@xlink:href: This is the filename of the file which will be extracted at installation, containing all the content of `<programlisting>` elements in this appendix.

The special value "cp" is supported in the @userlevel attribute for the handling of a Makefile, because the doc package cannot preserve TAB characters in an extracted document. Normally this does not matter too much, but in a Makefile, a TAB character is critical to the functionality of a recipe. Because of this, a documented, otherwise extractable Makefile in a *ClassPack* appendix with this "cp" value for the @userlevel *will not be extracted* and **MUST** be included in the Manifest as a static file (see [section 4.5 on page 46](#)).

3.7 Part "files": unannotated ancillary files

This part is for ancillary files which just need to be extracted whole, along with the class or package, and do not have any annotations.

These would typically be self-documenting example documents, default configuration files, shell scripts documented elsewhere, and other non- \LaTeX files.

Each one goes in a `<programlisting>` container element within a `<chapter>` which **MUST** give the filename to be created in the @xlink:href attribute, and **MUST** have a unique @xml:id to identify it within *ClassPack*.

The `<programlisting>` element MAY have its own `@xlink:href` attribute which points at an external file (eg a template) which is to be included at this point. In this case the `<programlisting/>` element is empty.

```
<part xml:id="files">
  <title>Ancillary files</title>
  <chapter xml:id="testdoc" xlink:href="testdoc.tex">
    <title>Test document</title>
    <para>You can use this document to test the class.</para>
    <programlisting language="LaTeXe">
%&lt;&lt;COMMENT
% !TEX TS-program = xelatex
% !TEX encoding = UTF-8 Unicode
% !BIB TS-program = biber
%COMMENT
    </programlisting>
    <programlisting language="LaTeXe">
\documentclass[12pt,a4paper]{myclass}
\begin{document}
Some testing stuff
\end{document}
    </programlisting>
  </chapter>
</part>
```

In the example, the \TeX editor signals block is included as a separate `<programlisting>` element *before* the one containing the code. This lets editors and other software identify which processors to use.

3.8 The README.md file

One additional user file is produced automatically by the xSLT scripts: the *Markdown* file called `README.md` which accompanies all classes and packages for display on CTAN. The default file contains a brief description of usage and installation, for the benefit of people who cannot or will not read the PDF documentation, with the package or class name and the Abstract of the documentation automatically inserted for the current document.

This is generated automatically from the file `readme.xml`, which is a *DocBook5* `<chapter>` document supplied as part of *ClassPack*. It contains some changes to the character entities to accommodate plain text, and some locations signalled for the replacement values of class or package name and the Abstract.

Note that this document does *not* use the special `doctexbook.dtd` used for your normal *ClassPack* XML documents; instead it uses the plain, unmodified *DocBook* which underlies the `doctexbook.dtd`, with a few entity declarations and a couple of extra attributes in the local subset.

The `readme.xml` file uses the `<olink>` element type to act as a placeholder for transcluded atomic information from the main document (pending implementation of the proposed *DocBook* Transclusions method). This element

MUST have a `<targetptr>` and a `<type>` attribute specifying (respectively) the name of the element type in the main document and the relevant attribute. For example, to include the name of the class or package, we use:

```
<olink targetptr="book" type="xml:id"/>
```

Two other element types have also had `<targetptr>` and `<type>` attributes added: `<sect1>` and `<anchor>`. These are used to specify the inclusion of whole sections or fragments of the main document, such as the Abstract or the Copyright.

The text is reformatted into *Markdown* by the templates in `db2md.xml`. Only a few element types have been implemented for this in this version: enough to convert the Abstracts of packages under the author's care and the `readme.xml` file itself.

The resulting `README.md` file therefore includes the Abstract from your class or package document as the first section. The `db2md.xml` script uses a template called `normtext` to create long single lines of *Markdown* text from elements such as titles and paragraphs. This handles the conversion of entities which occur in your Abstract (those declared in `doctexbook.dtd`): by default, `&TeX`;, `&LaTeX`;, and `&thinsp`; are catered for, but if you use any others, you must modify the code in this template to deal with them, using the nested `replace()` functions. The `db2md.xml` script handles all the element markup currently used in `readme.xml`, plus the `<para>` and `<literallayout>` elements used in the author's Abstracts. Any additional abstract elements required should be notified to the author.

3.9 Other output

Three other files are generated during processing:

MANIFEST: This is a list of all files in the package, *excluding* those which will be generated by the installation procedure in the `.ins` file. It is used to create the zip files, and it itself a part of the package.

autopack.xml: For the purposes of package tracking, the in-memory list of packages computed by the *AutoPackage* mechanism is serialised to an XML document in `autopack.xml`.

index.html: This is the sample entry for a web page about your package. Currently the formatting is used for the author's pages at <https://latex.silmaril.ie/packages/> but could easily be adapted in the `db2dtx` script (see [section 11.7 on page 290](#)).

3.10 Worked example

The file `template.xml` included in the package distribution contains examples of the structures in a *ClassPack* application, with comments to remind the author of what goes where.

3.10.1 Startup

The file starts with the XML Declaration followed by the Document Type Declaration for the modified version of *DocBook* (5) supplied with *ClassPack*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC
  "-//Silmaril//DTD DocBook 5.0 for DocTeX ClassPack//EN"
  "../doctexbook.dtd">
```

3.10.2 Root element specifications

The start-tag for the `<book>` element shows the attributes explained in [section 3.3 on page 21](#) which will need editing for each new application, and revising for updates, especially the `@version`, `@revision`, and `@security` attributes.

```
<book xml:id="" version="" xml:base=""
  remap="12pt" arch="class" audience="lpp1" condition="2017-04-15"
  conformance="LaTeX2e" os="all" revision="1" security=""
  userlevel="cls" vendor="" status="beta" annotations="\raggedright">
```

3.10.3 Packages for the documentation

A filename for artwork for a cover for the documentation can be specified in the optional `@coverart` attribute of the `<info>` element. The packages needed for your documentation (at least, those which do not get added automatically by *AutoPackage*) go in the `<seg>` element of `<seglistitem>` elements:

```
<info coverart="">
  <cover>
    <constraintdef xml:id="docpackages">
      <segmentedlist>
        <segtitle>Packages required for documentation
          (&XeLaTeX;)</segtitle>
        <seglistitem>
          <seg>fontspec</seg>
        </seglistitem>
        <seglistitem>
          <seg>parskip</seg>
        </seglistitem>
      </segmentedlist>
```

3.10.3.1 Ancillary startup commands for the documentation :

[DEPRECATED] The mechanism of earlier versions to provide for the computed construction of commands is no longer in use (see the warning in [section 11.4.1 on page 121](#), the code on [page 113](#), and the code on [page 121](#)). It is retained here for historical compatibility and will be removed in a future version.

```
<cmdsynopsis>
  <command>SIL@doctype</command>
  <arg remap="arch"/>
</cmdsynopsis>
<cmdsynopsis>
  <command>SIL@docname</command>
  <arg remap="xml:id"/>
</cmdsynopsis>
</constraintdef>
```

Code for the documentation which needs to be executed at the end of the Preamble but before the beginning of the document goes in a `<constraintdef>` element with the `@xml:id="atendpreamble"`. By this time all packages will normally have been loaded.

```
<constraintdef xml:id="atendpreamble">
  <procedure>
    <step>
      <cmdsynopsis>
        <command>\allsectionsfont{\sffamily}</command>
      </cmdsynopsis>
    </step>
  </procedure>
</constraintdef>
```

Code for the documentation which needs to be executed at the beginning of the document goes in a `<constraintdef>` element with the `@xml:id="atbegindoc"`. This will be executed *after* `\begin{document}`, that is, during the absorption phase of the `.dtx` file as it is run by the `\DocInput` command of the `doc` package.

```
<constraintdef xml:id="atbegindoc">
  <procedure>
    <step>
      <cmdsynopsis>
        <command>\setcounter{tocdepth}{5}</command>
        <command>\setcounter{secnumdepth}{5}</command>
        <command>\def\@doxdescribe#1#2{\endgroup
\ifdox@noprnt\else\marginpar{\raggedleft
\textcolor{DarkRed}{\@nameuse{PrintDescribe#1}{#2}}}\fi
\ifdox@noindex\else\@nameuse{Special#1Index}{#2}\fi
\endgroup\@esphack\ignorespaces}</command>
      </cmdsynopsis>
    </step>
  </procedure>
</constraintdef>
```

3.10.4 Packages for the class or style package

Packages required for your class or style package are specified in the same way as those for the documentation. Not that these packages do *not* get passed through *AutoPackage*: it is considered that the author is controlling the inclusion of packages manually as part of the design. The `@linkend` attribute on the `<constraintdef>` element **MUST** match a section in the code part: the package list will be output before or after that section, as determined by the value of the `@role` attribute.

```
<constraintdef xml:id="clspackages" linkend="classload" role="after">
  <segmentedlist>
    <segtitle>Packages required for the class or
      package</segtitle>
    <seglistitem>
      <seg>fontspec</seg>
    </seglistitem>
    <seglistitem>
      <seg>graphicx</seg>
```

```

    </seglistitem>
    <seglistitem>
      <seg role="svgnames">xcolor</seg>
    </seglistitem>
    <seglistitem>
      <seg role="margin=3cm">geometry</seg>
    </seglistitem>
  </segmentedlist>
</constraintdef>

```

3.10.5 The Manifest

The rules for package governance at CTAN require a file called MANIFEST containing a complete list of all files in the downloadable zip file submitted. *ClassPack* constructs this file automatically for those files it is managing directly, but this is the place to list any others that need including.

```

<constraintdef xml:id="manifest">
  <simplelist>
    <member></member>
  </simplelist>
</constraintdef>
</cover>

```

3.10.6 Metadata

The title of the document being created is generated automatically to be *The $\text{\LaTeX} 2_{\epsilon}$ package* (or ‘class’ as appropriate). The <title> needed here is in effect an explanatory subtitle, a one-line explanation of what this package or class is for. There can be multiple authors, in which case they MUST be enclosed in an <authorgroup> element. Keywords are recommended, as they also get used in the PDF metadata, which gets indexed by search engines. If the copyright date is given in ISO format in the @YYYY-MM-DD attribute, no content in the element is necessary. Both email address and name of the copyright holder SHOULD be given. The downloadable location of the package (eg CTAN or the author’s web site) SHOULD be given in the <releaseinfo> element.

```

<title></title>
<author>
  <personname>
    <firstname></firstname>
    <surname></surname>
  </personname>
  <affiliation>
    <orgname></orgname>
    <orgdiv></orgdiv>
  </affiliation>
  <address></address>
  <email></email>
  <uri></uri>
  <contrib role="sponsor"></contrib>
</author>
<keywordset>
  <keyword></keyword>
</keywordset>
<copyright>

```

```
<year YYYY-MM-DD=""></year>
<holder xlink:href="mailto:">Your Name</holder>
</copyright>
<releaseinfo></releaseinfo>
```

The admonition to report corrections SHOULD be left as-is.

```
<annotation role="UNUSED" xreflabel="">
  <para>This file was generated from an XML master source.
    Amendments and corrections should be notified to the
    maintainer for inclusion in future versions.</para>
</annotation>
```

The Abstract (renamed here ‘Summary’) MUST be short enough to fit about half of the front page of the documentation: two or three short paragraphs are plenty to explain what this class or package is and why you might want to use it. The optional image file in the `@xlink:href` attribute will be placed on the front page below the Summary.

```
<abstract xlink:href=".png">
  <title>Summary</title>
  <para></para>
</abstract>
```

The revision history is essential and must be completed after each change. The version number given MUST match the `@version` and `@revision` values on the document root `<book>` element. For minor changes, the `<revdescription>` element content can be a single `<para>` element.

```
<revhistory xml:id="changehistory">
  <!-- the latest version must match the root element
       @version and @revision attributes, eg 3.14 -->
  <revision version="0.1">
    <date YYYY-MM-DD=""/>
    <revdescription>
      <itemizedlist>
        <title></title>
        <listitem>
          <para></para>
        </listitem>
      </itemizedlist>
    </revdescription>
  </revision>
</revhistory>
</info>
```

3.10.7 Documentation

The ID “doc” is MANDATORY. The `@conformance` MUST give the name of the \LaTeX executable to be used (*xelatex*, *lualatex*, or *pdflatex*). Beyond that, you write your documentation in *DocBook*, using the supported markup as described in [section 6 on page 56](#).

```
<part xml:id="doc" conformance="xelatex">
```

```

<title></title>
<chapter>
  <title>Introduction</title>
  <para></para>
</chapter>

```

Bibliographic references follow the documentation. Each entry MUST be in a `<biblientry>` element.

```

<bibliography label="apa" xreflabel="" arch="bibtex" condition="biber">
  <!-- one of (bibliodiv bibliomixed biblientry) -->
</bibliography>
</part>

```

3.10.8 Code

The first `<chapter>` of the code MUST give the name of the file to which it is to be extracted from the `.dtx` file. This will be the package name with the file extension `.cls` or `.sty` as appropriate. This must match the computed name derived from the name of the application directory plus the filetype. Code can be structured in any suitable way: an "options" section would be common...

```

<part xml:id="code">
  <!-- @xlink:href is the name of the file to generate -->
  <chapter xml:id="" xlink:href="">
    <title>Implementation</title>
    <para></para>
    <sect1 xml:id="options">
      <title>Options</title>
      <para></para>
      <annotation role="option" xreflabel="name">
        <para></para>
      </annotation>
    </sect1>
  </chapter>
</part>

```

...as would one with an ID that can be matched by the value of the `@linkend` attribute on the `<constraintdef>` element with the `@xml:id="clspackages"` (or "stypackages" as appropriate) as shown in [section 3.10.4 on page 34](#). The base class (if relevant) can be given in the `\PassOptionsToClass` and `\LoadClass` commands; otherwise that code block can be omitted. The use of the `<annotation>` element is described in [section 3.6 on page 27](#).

```

<sect1 xml:id="classload">
  <title>Load the document base class</title>
  <para>This class is based on the standard &LaTeX;
    <classname>report</classname> class, with no special
    options.</para>
  <programlisting>
\DeclareOption*{%
  \PassOptionsToClass{%
    \CurrentOption}{report}}
\ProcessOptions\relax
\LoadClass{report}
  </programlisting>
</sect1>

```

```

    <!-- packages get loaded here in this example -->
  </sect1>
  <sect1 xml:id="packagemods">
    <title>Changes to package defaults</title>
    <para></para>
    <annotation role="macro" xreflabel="name">
      <para></para>
      <programlisting>
      </programlisting>
      <para></para>
    </annotation>
  </sect1>
</chapter>

```

3.10.9 Extractable *annotated* files

Files for extraction that are annotated with documentation go in `<appendix>` elements within the code part. As with the main class or package, the code can be inline within the XML document, or referenced by line numbers or by fragments created by the *chunk* (1) utility supplied (see item 3 in the list on p. 61).

```

<appendix xml:id="" xlink:href="output">
  <title></title>
  <para></para>
  <programlisting language="" xlink:href="input"/>
</appendix>
</part>

```

3.10.10 Extractable files without annotation

Files for extraction that are supplied as-is MUST go in `<chapter>` elements in the files part. The filename to extract *to* MUST be given in the `@xlink:href` of the `<chapter>` element. If the files are to be brought in from an external source rather than supplied inline, that file name MUST go in the `@xlink:href` on the `<programlisting>` element.

```

<part xml:id="files">
  <chapter xml:id="" xlink:href="">
    <title></title>
    <para></para>
    <programlisting language="" xlink:href=""/>
  </chapter>
</part>
</book>

```

4 Package management

As mentioned above, the `<cover>` element is used to provide places where packages and other \LaTeX preliminaries can be specified. Using this structure means each entry is separately editable, and the same structure, with different content, is used both for packages for the documentation *and* packages for the class or package being created.

It would of course have been possible just to allow a slab of `\usepackage` commands at these points, but that would have made commenting and documentation harder, and would also have made it more difficult to perform an XML editor's element-copy-element-paste or an `XInclude` when using one package's or class's settings as the basis for another.

However, the most important reason is that specifying package lists as separately-identifiable values makes it possible to automate the documentation of frequently-used packages, and to automate the inclusion of parameters, options, settings, additional commands, and even documentation, which you can store separately for re-use in each class or package you maintain by adding your own modifications that you like to have included whenever you use a particular package.

The *AutoPackage* feature added to *ClassPack* in v0.74 means that packages needed for your documentation on a regular basis can now be detected automatically on the basis of features you use in your documentation, making it unnecessary to specify them by hand at all: full details are in [section 5 on page 48](#). For example, if you use compact lists (*DocBook's* `spacing="compact"` attribute), *ClassPack* will detect this and add the `enumitem` package so that \LaTeX can implement compact spacing.

The *db2dtx* script also uses this feature in order to modify the behaviour of standard \LaTeX at several points (such as fixing the broken abstract formatting when the `parskip` package is used, and fixing the broken formatting of long item label values in the `description` environment).

It is important to note that *AutoPackage* applies *only* to the selection of packages for your documentation. Although the same XML structure is used to hold the package list for your class or style package, it leaves the selection to you as the author because there is no text on which to base decisions about inclusion or exclusion.

4.1 Where to put the lists of packages

Packages you need **MUST** be specified in the `<cover>` element in a `<constraintdef>` element with one of the `@xml:id` values `"docpackages"`, `"clspackages"`, or `"stypackages"`.

At the moment, only one of the `"clspackages"` or `"stypackages"` lists can be used in a single *ClassPack* document, because you are either creating a class *or* a style package, not both.

```

<constraintdef xml:id="docpackages">
  ...packages needed for the user documentation go here...
</constraintdef>

<constraintdef xml:id="clspackages" linkend="options">
  ...packages needed for the class you are writing go here...
</constraintdef>

<constraintdef xml:id="stypackages" linkend="options">
  ...packages needed for the package you are writing go here...
</constraintdef>

```

The exact @xml:id values given are MANDATORY on the relevant <constraintdef> elements: the @xml:id values are used by *db2dtx*, matching the first three letters with the three-letter filetype used as the value of the @arch attribute that you specified on the <book> root element; see ‘@arch’, the second item in the list in [section 3.3 on page 22](#)).

Note that there are two other, unrelated, uses for the <constraintdef> element, dealt with in [section 4.4 on page 45](#) and [section 4.5 on page 46](#), which must not be confused with this one.

4.2 Listing the packages

The method is identical for all three cases: each <constraintdef> element holds a <segmentedlist> to specify the packages required. It MUST have a title to label it.

Within the list, each package goes in a <seglistitem> element in a <seg> subelement.

```

<constraintdef xml:id="docpackages">
  <segmentedlist>
    <segtitle>Packages required for documentation</segtitle>
    <seglistitem role="Use the Charter typeface for documentation.">
      <seg version="2005-04-12">charter</seg>
    </seglistitem>
    <seglistitem role="Use Helvetica as the sans-serif, but scale it to fit">
      <seg role="scaled=0.8333">helvet</seg>
    </seglistitem>
    <seglistitem>
      <seg role="svgnames">xcolor</seg>
    </seglistitem>
    <seglistitem>
      <seg>array</seg>
    </seglistitem>
    ...
  </segmentedlist>
</constraintdef>

```

The rules are:

- As noted earlier, of the two values “`clspackages`” and “`stypackages`”, only one can be used in a single *ClassPack* document.

This will be fixed in a later version, as it interferes with the generation of additional package files to accompany a single class.

- An `<segtitle>` element **MUST** start each such list. It will be used as a comment (for the documentation) or a subheading (classes or packages);
- Each `<seglistitem>` **MAY** have a documentary comment in the `@role` attribute about why this package is being used. In the case of the packages for your own class or package, this comment is reproduced in the documentation of the code.

If an *AutoPackage* included package is specified, this comment is included *after* any documentation that gets added by the *AutoPackage* mechanism (see [section 5.4 on page 52](#) for details);

- If the package must conform to a specific version, the date **MUST** be provided in ISO 8601 format in the `@version` attribute;
- Where a package will be included automatically with specific options by the *AutoPackage* mechanism, and you want to override them, give the specification you want in the `@role` attribute of the `<seg>` element as normal, but set the `@condition` attribute to “only”, and ensure that there is an entry for the specification you want in `prepost.xml` positioned earlier than the one which would otherwise have been autoincluded. Currently the only candidate for this is the `inputenc` package, for which `utf8x` would be the normal default; but its inclusion with the `utf8` (no x) is achieved by this method;
- If you want the package you are maintaining to be included in its own documentation (perhaps so you can use it for examples), you do it by setting the `@xlink:role` attribute on the `<book>` root element to null as described in ‘`@xlink:role`’, the last item in the list in [section 3.3 on page 23](#). You do *not* include it in the list of packages described here. If you want specific options, they can be given as the content of the `@xlink:role` attribute.

Order is significant *except* for packages for which there is an entry in the `prepost.xml` database file (see [section 5.4 on page 52](#) for details). Those matching packages are always invoked first, *in the order in which they occur in `prepost.xml`*, and any remaining (unmatched) packages come afterwards. The `dox` package is a special case: it always comes first.

The resulting order is therefore: *a)* `dox`; *b)* packages requested in the *Classpack* document which match `prepost.xml` packages, merged and sorted in the order in which they occur in `prepost.xml`; and *c)* any other packages requested in the *Classpack* document. For example, for this document (*ClassPack*’s own documentation), `classpack.xml` says:

```
<constraintdef xml:id="docpackages">
  <segmentedlist>
    <segtitle>Packages required for the documentation</segtitle>
    <seglistitem>
```

```

        <seg role="ragged">footmisc</seg>
    </seglistitem>
    <seglistitem>
        <seg>bbding</seg>
    </seglistitem>
    <seglistitem>
        <seg role="obeyspaces,spaces">url</seg>
    </seglistitem>
</segmentedlist>
</constraintdef>

```

The console log of the process shows what was actually included: first the results of the package-detection phase, showing why each package was being selected:

```

This is DB2DTX, Version 1.27 (2024-01-30).
Making index.html for web site
Making package classpack
dox (1.001) default
fix-cm (1.002) attribute-match=1 type/
fontspec (1.003) attribute-value match=1 part/@conformance=xelatex:
fontenc (1.006) default will be omitted because of fontspec
inputenc (1.007) attribute-value match=1 bibliography/@arch=biblatex:
inputenc (1.009) default will be omitted because of fontspec
noto (1.019) default
bbding (1.038) element-match=176 acronym
mflogo (1.041) default
babel (1.044) default
biblatex (1.045) element-match=154 biblioref,exceptionname,citetitle
biblatex (1.045) attribute-value match=1 bibliography/@arch=biblatex:
csquotes (1.047) attribute-value match=1 bibliography/@arch=biblatex:
array (1.051) default
calc (1.057) default
ccaption (1.060) default
enumitem (1.065) element-match=31 variablelist
enumitem (1.065) attribute-value match=6 orderedlist/@spacing=compact:@spacing=compact:,
    itemizedlist/@spacing=compact:@spacing=compact:@spacing=compact:@spacing=compact:
fancybox (1.066) element-match=14 note
fancyvrb (1.068) element-match=1 bibliography
relsize (1.073) element-match=176 acronym
textcase (1.076) default
float (1.078) default
fmtcount (1.079) attribute-value-pointer (37) xref/→@linkendvarlistentry; →@linkendlistitem;
geometry (1.080) default
fancyhdr (1.082) default
graphicx (1.086) attribute-value match=1 part/@conformance=xelatex:
listings (1.092) element-match=73 programlisting
ltxcmds (1.094) default
makeidx (1.095) default
parskip (1.104) default
ragged2e (1.106) attribute-value match=1 bibliography/@remap=shortbib:
sectsty (1.107) default
ulem (1.114) element-match=3 link
url (1.115) element-match=12 email,uri,link
varioref (1.118) element-match=139 xref
xcolor (1.126) attribute-match=1 type/
xcolor (1.126) dependency match dox hypdoc hyperref classpack
classpack (1.130) default
hypdoc (1.132) default
hyperref (1.133) dependency match hypdoc
menukeys (1.134) element-match=5 guibutton,guilabel,guimenu,guisubmenu,guimenuitem
Passing options "svgnames" to package xcolor
Passing options "obeyspaces,spaces" to package url

```

In the resolution phase, packages are sorted in the order determined in the `prepost.xml` file, with multiple detections collapsed to a single `\usepackage`

command.

```

1.001. Package 'dox' (1×) being included
1.002. Package 'fix-cm' (1×) being included
1.003. Package 'fontspec' (1×) being included
1.006. Package 'fontenc' (1×) omitted: transcluded by fontspec
1.007. Package 'inputenc' (1×) omitted: transcluded by fontspec
1.009. Package 'inputenc' (1×) omitted: transcluded by fontspec
1.019. Package 'noto' (1×) being included
1.038. Package 'bbding' (2×) being included
1.041. Package 'mflogo' (1×) being included
1.044. Package 'babel' (1×) being included
1.045. Package 'biblatex' (2×) being included
1.047. Package 'csquotes' (1×) being included
1.051. Package 'array' (1×) being included
1.057. Package 'calc' (1×) being included
1.060. Package 'ccaption' (1×) being included
1.065. Package 'enumitem' (2×) being included
1.066. Package 'fancybox' (1×) being included
1.068. Package 'fancyvrb' (1×) being included
1.073. Package 'relsize' (1×) being included
1.076. Package 'textcase' (1×) being included
1.078. Package 'float' (1×) being included
1.079. Package 'fmtcount' (1×) being included
1.080. Package 'geometry' (1×) being included
1.082. Package 'fancyhdr' (1×) being included
1.086. Package 'graphicx' (1×) being included
1.092. Package 'listings' (1×) being included
1.094. Package 'ltxcmds' (1×) being included
1.095. Package 'makeidx' (1×) being included
1.104. Package 'parskip' (1×) being included
1.106. Package 'ragged2e' (1×) being included
1.107. Package 'sectsty' (1×) being included
1.114. Package 'ulem' (1×) being included
1.115. Package 'url' (2×) being included
1.118. Package 'varioref' (1×) being included
1.126. Package 'xcolor' (2×) being included
1.130. Package 'classpack' (1×) being included
1.132. Package 'hypdoc' (1×) being included
1.133. Package 'hyperref' (1×) being included
1.134. Package 'menukeys' (1×) being included
1.069. Package 'footmisc' (1×) being included
Creating package 'classpack' (v1.27) with checksum 2409.

```

Some of these packages are included by default because I have specified in my personal copy of the `prepost.xml` database that they should be, because I tend to use them in all my packages. Others are included because I have used some document feature, for example `<tgroup>` for specifying a table layout implies that I will need the array package. This makes it much easier just to *write*, and let *ClassPack* take care of using the right packages.

New users should modify the `prepost.xml` database contents and sort order to suit their particular style and way of working.

4.3 Specifying where in the .dtx file to output packages for a class or package

Because the list of packages is stored out of line with respect to your class or package code, *ClassPack* needs to be told *where* the relevant `\RequirePackage` commands must be added to the .dtx file. Your class or package design may also include preliminary commands needed before packages are included or before a base class is loaded. You specify the place to include packages by using the `@linkend` attribute on the `<constraintdef>` element to give the value of an `@xml:id` on a particular `<chapter>` or `<sect1>` in your annotated code.

There is no default: you MUST specify this link yourself, otherwise the list of required packages will not be output.

- If the `<chapter>` or `<sect1>` which you specify has content (text) in it, the `\RequirePackage` commands are output as the content of a new `<chapter>` or `<sect1>` immediately preceding or following it, as specified by the value of the `<role>` attribute on the same `<constraintdef>` element ("before" or "after").
- If the `<chapter>` or `<sect1>` you have specified is empty (no text content), the `\RequirePackage` commands are output *as if they were the content* of that `<chapter>` or `<sect1>`.

As an example, suppose you specify the `<constraintdef>` with

```
<constraintdef xml:id="clspackages" linkend="options" role="after">
...
</constraintdef>
```

There MUST then be a `<chapter>` or `<sect1>` in your annotated code with the `@xml:id` value of "options". If it is empty of any text (no character data content) like this:

```
<sect1 xml:id="options">
  <title/>
  <para/>
</sect1>
```

then the list of `\RequirePackage` commands will be output *in its place*. (Note that the empty title and paragraph are required by *DocBook* for validity.)

If, on the other hand, the specified section has text and code of its own:

```
<sect1 xml:id="options">
  <title>Options</title>
  <para>text...</para>
  <programlisting>
\some{code}
  </programlisting>
</sect1>
```

then the list of `\RequirePackage` commands will be output immediately after it, as a new `<sect1>`.

In both cases, the `<segtitle>` of the `<segmentedlist>` whose packages are output at this point will be used as the title of the section.

4.4 Additional setup commands for documentation

In the documentation for a class or package, there is sometimes a need to use individual \LaTeX commands to adjust document settings more finely than a package provides, either in the Preamble of the `.dtx` file or even immediately after the `\begin{document}` command.

There would typically not be many of these, as the default format for Doc \TeX documents is the (fairly plain) `ltxdoc` class, based on the `article` class, and rarely varied. In the case where an in-house documentation style is needed, or an author prefers to use a different style with the `ltxdoc` class, the changes would normally be implemented in a separate package, so there would equally not be many individual commands to implement here.

When there is a need for setup commands, they go in *additional* `<constraintdef>` elements (this is the first of two additional use cases mentioned in the warning in [section 4.1 on page 40](#): the other is in [section 4.5 on the following page](#)).

Note that commands that you want implemented *every time you use a particular package* should go in your `prepost.xml` database file, as described in [section 5 on page 48](#).

4.4.1 \LaTeX commands to appear in the documentation Preamble

Use the `@xml:id` value of `"atendpreamble"` on the `<constraintdef>` element. Commands are output to the `.dtx` file immediately after the documentation's `\usepackage` commands.

```
<constraintdef xml:id="atendpreamble">
  <procedure>
    <step>
      <cmdsynopsis>
        <command>\setmonofont[Scale=MatchLowercase]{Inconsolata}</command>
      </step>
    </procedure>
  </constraintdef>
```

In this case, the `Inconsolata` font is not available as a package, so a separate command is needed.

4.4.2 \LaTeX commands to appear before the start of the documentation text

If the commands have to go after the Preamble, set the `@xml:id` attribute to `"atbeginndoc"` on the `<constraintdef>` element. Commands in this category are output at the start of the documentation text *after* the `\begin{document}`

command (where they will be executed during the `\DocInput` phase of DocTeX processing). For example:

```
<constraintdef xml:id="atbegindoc">
  <procedure>
    <step>
      <cmdsynopsis>
        <command>\setcounter{tocdepth}{5}</command>
        <command>\setcounter{secnumdepth}{5}</command>
        <command>\def\@@doxdescribe#1#2{\endgroup
\ifdox@noprint\else\marginpar{\raggedleft
\@nameuse{PrintDescribe#1}{\LabelFont[\color{DarkRed}]{#2}}}\fi
\ifdox@noindex\else
\@nameuse{Special#1Index}{#2}\fi
\endgroup\@espack\ignorespaces}</command>
      </cmdsynopsis>
    </step>
  </procedure>
</constraintdef>
```

The new `dtxdescribe` package now provides much of this functionality; A future version of `ClassPack` will drop this code and switch to using `dtxdescribe`.

(The `\@@doxdescribe` command is an oddity here: all it does is redefine the marginal print value for ibline documentation to use a narrow red font for reference purposes, but it does not work if placed in the `prepost.xml` database file.)

Commands specified in this category are automatically tested for the presence of @ signs, and enclosed in `\makeatletter` and `\makeatother` commands.

4.5 The Manifest

The second variant of the `<constraintdef>` element (with the `@xml:id` attribute of "manifest") is very simple. It is a list of the names of any separate files that you want included in the Zip file that the `make` command produces. This means anything other than the `.dtx`, `.ins`, `.pdf`, and `README.md` files that get included automatically.

The content of this element is a `<simplelist>`, containing `<member>` elements, one per file, for example:

```
<constraintdef xml:id="manifest">
  <simplelist>
    <member>doctexbook.dtd</member>
    <member>db2dtx.xsl</member>
    <member>db2bibtex.xsl</member>
    <member>figtab2latex.xsl</member>
    <member>db2md.xsl</member>
    <member>prepost.xml</member>
    <member>lppl.xml</member>
    <member>decommentbbl.awk</member>
```

```
<member>readme.xml</member>  
<member>Makefile</member>  
</simplelist>  
</constraintdef>
```

5 Automated package inclusion

There are several reasons for automating the inclusion of packages, especially in documentation:

- Many \LaTeX authors and maintainers have favourite packages or settings that they like to use every time they write class or package documentation.
- Some options have now become the *de facto* convention for their package (for example, the `utf8x` option on the `inputenc` package and the `T1` option on the `fontenc` package);
- Authors and maintainers may have their own per-package preferences to suit their way of writing.
- There are *de jure* commands that have to be used whenever a particular package is invoked (for example, using the `makeidx` package means you **MUST** add the `\makeindex` command to the Preamble).
- Some packages depend on aspects of the documentation; that is, they only become necessary when a particular formatting feature is used (for example certain tabular alignment column types like `m` or `b` **REQUIRE** the `array` package; or using `<itemizedlist>` or `<orderedlist>` *inside* a paragraph **REQUIRES** the `enumitem` package, which enables inline lists with automated punctuation). This avoids the author having to remember to include a specific package when such a feature is used, or to remember to remove it if the feature is no longer used.
- Some packages are *dependencies* on other packages. Most of these are handled within those packages, but (for example) if you want a specific colour palette option like `svgnames` in the `xcolor` package when using `hyperref`, you need to signal this to `xcolor` beforehand with a `\PassOptionsToPackage` command, as in [section 5.1](#).

AutoPackage uses the XML structure of the document to detect when specific packages are needed, via the `prepost.xml` file, which acts as a lookup.

5.1 Packages preloaded before the document class declaration

One package (`fix-cm`) is invariably preloaded *before* the document class (`ltxdoc`) is declared, and two further packages have an option pre-specified with the `\PassOptionsToPackage` in certain circumstances. These are currently:

```
\RequirePackage{fix-cm}
\PassOptionsToPackage{svgnames}{xcolor}
\PassOptionsToPackage{british}{babel}
```

- The `fix-cm` package removes the lock on using *only* the predefined step-sizes of type specified in the Font Definition files for Computer Modern (and followed for many other fonts).²

²This may be replaced at a future date with the `anyfontsize` package.

This package allows arbitrary font sizes to be specified, which is useful — even essential — in the documentation of code blocks (in the `<programlisting>` element), where a block with an unavoidably long line may need a carefully-chosen smaller font size to fit. This is required even with \LaTeX . See `@wordsize`, the second item in the list in [2 on page 61](#) for details of how to do this;

- We prefer to use the SVG colour palette by default; the `hyperref` package does not, making it hard to respecify a palette once it is loaded. Passing the option to `xcolor` in advance overrides this problem. This option is specified by default, because `xcolor` is required by `hyperref`, which we use by default as it gets loaded by `hypdoc`; and also by *ClassPack*'s modifications to the `dox` package. This option should probably be configurable.
- The `url` package used to be preloaded with the `hyphens` option, but this code was removed in v1.16 because it was felt to be problematic as a piece of automation (it applies to breaking on hyphens in URIs, which is deprecated).
- The `babel` package used to be loaded automatically from `prepost.xml` with the single option `british` as a result of using `biblatex`, and then loaded again if multiple languages were detected. This led to option clashes, so now the multiple languages are detected from the use of the `@xml:lang` attribute in the document, and the relevant `\PassOptionsToPackage` command created. If there were no `@xml:lang` attributes, this command is not issued.

The default value for English-language \LaTeX is `en-US` (American English). To change this, specify a new value in the `@xml:lang` attribute on the document root element, `<book>`. Note that the order of entries in the `languages.xml` file is *significant* and that where there is ambiguity, the *last* matching value wins.

These features are currently all implemented in the XSLT code: a future version may move them to a configuration file, perhaps `prepost.xml`.

5.2 The `prepost.xml` lookup file

To automate package loading, an ancillary lookup file `prepost.xml` is used as a database. This file is a *DocBook* document with a `<refsection>` root element type containing exactly two `<procedure>` elements, as shown below.

The location of the `prepost.xml` file is specified in the `$(CPP)` variable in the `Makefile`. It would typically be a shared copy, located in the parent directory (that is, one level above the directories used for individual class and package projects), but it MAY equally be a per-project file in the project directories themselves; so long as the `Makefile` points at the right copy, all shall be well.

```
<refsection>
  <title>Commands to use before and after packages being loaded</title>
  <procedure xml:id="prepackage">
    ...steps...
  </procedure>
  <procedure xml:id="postpackage">
    ...steps...
```

```
</procedure>
</refsection>
```

1. Each entry for a package goes in a `<step>` element with the `@remap` attribute set to the name of the package, and the `@condition` attribute containing a space-separated list of use cases (`doc`, `cls`, and `sty`). See the example below and the description in [section 5.3 on the following page](#).
2. A package MAY be set up in either `<procedure>` section, or both.
 - The “prepackage” section is for packages with additional commands which need to be executed *before* the package is invoked.
 - The “postpackage” section is for packages with additional commands which need to be executed *after* the package is invoked.
3. If a package requires neither type of additional command[s], it MAY be put in either `<procedure>` section.
4. An entry for a package SHOULD also contain a sentence of documentation in a `<para>` subelement, which will be used in the documentation of the code.
5. Order is *critical*: the order in which packages are written to the `.dtx` file (and therefore the order in which they are loaded in \LaTeX) is determined by the order of the `<step>` elements in this file.

This frees the author from having to consider what order to specify package in, but it imposes a duty on the *ClassPack* maintainers to keep the order of `<step>` elements correct.

Here is a simple example of an entry for the `bbding` package for assorted dingbats, icons, and symbols:

```
<step condition="doc cls sty" remap="bbding">
  <para>Collection of dingbats (icons).</para>
</step>
```

This specifies that the `dingbats` package may be requested by the author and it may be loaded for documentation, a class, or a package; and that the specified sentence of documentation will be included in the documentation, in the list of packages loaded.³

Packages are output to the `.dtx` file using `\RequirePackage` commands (for the documentation) or `\usepackage` commands (for the class or package itself), *in the order in which they occur in the `prepost.xml` file, regardless of the order in which they were given in the document* (that it, in the `<segmentedlist>` element where they were specified: see [section 4.2 on page 40](#)). The console log shows the entry number in `prepost.xml` for reference, and this also appears in the `autopack.xml` log file in the `@userlevel` attribute (see ‘`autopack.xml`’, the second item in the list in [section 3.9 on page 32](#)).

³Apart from the sentence of documentation, this entry is largely redundant, as any package requested by an author, and not present in `prepost.xml`, will be included anyway.

This allows `prepost.xml` to impose a known required sequence of some packages which **MUST** be loaded before others — for example, `hyperref` being loaded last of all, except for glossaries, which needs to come after that; and font packages under \LaTeX or \LuaTeX , which may need a specific order of invocation.

Scaling fonts with \LaTeX and \LuaTeX

Some sans-serif and monospace fonts may need scaling to match the document's serif font, so they **MUST** be loaded *after* the serif font. Font packages in `prepost.xml` are therefore grouped using the `@arch` attribute: "serif", "sans", "mono", "math", "hand", and "symbol". The scaling can be done with the option `Scale=MatchLowercase` in the `@role` attribute of the `<step>` for the package.

The exception to this ordering is packages you specify in the document for which no entry in `prepost.xml` [yet] exists. These are loaded last of all, which is a known risk. A warning message is issued to the console during the processing of the XSLT script if a package is specified which has no entry in `prepost.xml`, and they are shown with the number 666.

We **STRONGLY RECOMMEND** developers to add all missing packages which they use to their local copy of `prepost.xml`; to document and pay careful attention to where in the file they locate them; and periodically to upload the file to Silmaril so that others can benefit from their research. The `@vendor` attribute can be used to add the developer's username or number so that acknowledgement can be given.

5.3 Specifying known packages and default options

Within the two `<procedure>` elements, each package is identified in a `<step>` element.

- the `@remap` attribute holds the package name;
- the `@condition` attribute holds the type[s] of use this entry is intended to be effective for, "doc", "cls", or "sty" (space-separated if more than one);
- the optional `@role` attribute holds any default options (comma-separated, exactly as they would be if typed into a \LaTeX document) that should be applied when the package is invoked;
- the optional `@revision` attribute can hold a date (in ISO 8601, not \LaTeX format) used to impose a version-date restriction on the loading of the package. Note that a date specified for a package listed in a `<seglstitem>` in the *ClassPack* document itself will override any date specified in the `prepost.xml` file;
- the optional `@arch` used to identify the class of a font package: "serif", "sans", "mono", or "math" or "hand" or "symbol" (see the note in [section 5.2](#)).
- the optional `@conformance` attribute can hold a coded warning that some additional processing is needed. Only one value is currently supported,

"nosc", to signal that a font has no Caps and Small Caps variant; so that if the document markup invokes a feature known to require small caps, an alternative can be provided. This is experimental and may be dropped or superseded.

The `<step>` element MUST contain a `<para>` subelement holding a short explanation of what the package does. This is added as a comment to the annotated code, when invoked by a package listed in a `<seglistitem>` in the "stypackages" or "clspackages" `<constraintdef>` settings (see [section 4.2 on page 40](#)).

```
<step role="utf8x" remap="inputenc" condition="cls doc sty"
      revision="2008-03-30">
  <para>UTF-8 is the default character set, to allow for use of
    any character in any writing system. Some characters
    are not specified for all fonts, so may have to be
    specified manually.</para>
</step>
```

In the example above, specifying `inputenc` in a *ClassPack* document, in a `<seg>` element as described in [section 4.2 on page 40](#), results in the package being added to the class or package code with the option `utf8x` and the date constraint:

```
\RequirePackage[utf8x]{inputenc}[2008/03/30]
```

A `<step>` element may occur in the `prepost.xml` file up to three times in each `<procedure>` section in `prepost.xml`, with different settings (one for a class, one for a package, and one for the documentation). In the post-package section, the package would only appear to specify command code to include *after* the package gets loaded, not for any other purpose. A documentation paragraph only need occur once, even if the package is entered both in the pre- and post-package sections, but in the post-package section, the paragraph typically describes the additional commands that are being included.

5.4 Automating inclusion

To make a package load automatically into every class or package or documentation, according to specified conditions, add one or more `<constructorsynopsis>` elements within the `<step>` element for a package (*after* the `<para>` element) *in the pre-package section*.

```
<step condition="doc" remap="varioref">
  <para>Provides extended formatting of cross-references.</para>
  <constructorsynopsis condition="xref"/>
</step>
```

These `<constructorsynopsis>` elements specify the condition(s) under which the package will automatically be included *without the author needing to specify it* in a `<seg>` element in the document itself (as described in [section 4.2 on page 40](#)).

The `@condition` attribute on the `<constructorsynopsis>` element specifies a *DocBook* XML element type name. If this element type occurs anywhere in the

relevant <part> of the *ClassPack* document (as specified in the @condition attribute of the parent <step>), the package is included.

In the example above, the varioref package will be added if an <xref> element is present anywhere in the documentation.

There are six ways in which a package can be specified for inclusion:

5.4.1 Default inclusion (unconditional)

If the @condition attribute is present but empty (set to ""), the package will always be included in the type of output ("doc", "cls", or "sty") specified in the parent <step> element start-tag.

5.4.2 Inclusion when a particular element type is used

If the @condition attribute holds the name of a *DocBook* element type, the package will be included automatically if this element type is used anywhere in the documentation part of the *ClassPack* document (see example above).

5.4.3 Inclusion when an attribute on an element type is used

If one or more <methodparam> subelements are present, they specify attribute conditions on the element type named in the @condition attribute.

```
<step condition="cls sty" remap="graphicx">
  <para>Provide for graphics (PNG, JPG, or PDF format (only) for
    pdflatex; EPS format (only) for standard &LaTeX;).</para>
  <constructorsynopsis condition="imagedata">
    <methodparam>
      <parameter>fileref</parameter>
    </methodparam>
  </constructorsynopsis>
</step>
```

The <parameter> element specifies the name of an attribute. If no <modifier> element is present, the package will be included automatically if the attribute is present in the document on the specified element type, regardless of its value.

In the example above, the graphicx package will be included if the class or style package code includes an <imagedata> element with a @fileref attribute.

5.4.4 Inclusion when an attribute on an element type has a specific value

A <modifier> element specifies a value for the attribute named in the <parameter> element; the package will be included automatically if the attribute is used with this value in the document on the specified element type.

```
<step remap="dcolumn" condition="doc">
  <constructorsynopsis condition="colspec">
    <methodparam>
      <parameter>align</parameter>
      <modifier>char</modifier>
    </methodparam>
  </constructorsynopsis>
</step>
```

In the example above, a `<colspec>` element anywhere in the documentation with an `@align` attribute equal to `"char"` will cause the `dcolumn` package to be included automatically (that package handles decimal-column alignment).

In this version of *ClassPack*, multiple instances of the `<methodparam>` selector are treated as alternatives: the package will be included if just one of them is true. There is no provision at the moment for applying a Boolean AND to this selection mechanism.

5.4.5 Inclusion when an element references a specific element type via the ID/IDREF mechanism

This is a special case involving the use of the `@funcparams` subelement *instead of* the `@parameter` attribute. This specifies that an IDREF attribute must be checked for the *type* of element it refers to.

```
<step condition="doc" remap="fmtcount">
  <constructorsynopsis condition="xref">
    <methodparam>
      <funcparams>linkend</funcparams>
      <modifier>varlistentry</modifier>
    </methodparam>
    <methodparam>
      <funcparams>linkend</funcparams>
      <modifier>listitem</modifier>
    </methodparam>
  </constructorsynopsis>
</step>
```

In this example, the `fmtcount` package will be included if there is an `<xref>` element anywhere in the documentation with a `@linkend` attribute which points at an *element type* of `<varlistentry>` or `<listitem>` (rather than an element with a specified value, as would be the case with using `<parameter>`); that is, if there is a `<varlistentry>` or `<listitem>` with an `@xml:id` attribute which is referenced from some `<xref>` element's `@linkend` value, the `fmtcount` package will be included (that package enables ordinal counting, needed when making a reference to an item in a list that is not numbered, or when counting the number of items for a `<coref>` element type: see '`<coref>`', the seventh item in the list in [section 6.4 on page 66](#)).

5.4.6 Restricting the loading of conflicting packages

If the `<constructorsynopsis>` element has a `@remap` value containing one or more package names (separated by spaces), then the autoloading of this package will *prevent* the autoloading of the named packages, *even if they have been requested by the author*. This is to prevent known conflicts, of which the following is a good example:

```
<step condition="doc" remap="fontspec">
  <para>Font specification setup for use with &XeLaTeX;.</para>
  <constructorsynopsis remap="inputenc fontenc" condition="part">
    <methodparam>
      <parameter>conformance</parameter>
      <modifier>xelatex</modifier>
    </methodparam>
```

```
</constructorsynopsis>
</step>
```

If you have specified to use \LaTeX for the documentation, by setting the `@conformance` attribute on the "doc" `<part>` element to "xelatex", the `fontspec` package will be autoloaded, but the `inputenc` and `fontenc` packages will not be, even if you have requested them, because you cannot use them in the same document as `fontenc`.

5.5 Adding setup commands before or after a package

Each `<step>` may also contain one or more `<constraintdef>` elements containing `<cmdsynopsis>` elements containing `<command>` elements to hold \LaTeX code to be inserted, in exactly the same format as shown in [section 4 on page 39](#).

```
<step remap="apacite" condition="doc">
  <constraintdef>
    <cmdsynopsis>
      <command>\AtBeginDocument{\edef\ApaciteRestoreAtCode%
        {\catcode`\@=\the\catcode`\@relax}}</command>
    </cmdsynopsis>
  </constraintdef>
</step>
```

- If the step is given in the "prepackage" section, the code is inserted *before* the package is invoked;
- if the step is given in the "postpackage" section, the code is inserted *after* the package is invoked;

that is, before or after the `\RequirePackage` or `\usepackage` command is output.

6 Using *ClassPack* to write the documentation

The text of your documentation goes in a `<part>` element with the `@xml:id` attribute set to "doc".

The tag-set of *DocBook* is very large, and only a part of it is needed for this purpose, although support for additional elements is easily added in the XSLT script.

The following sections describe the elements and attributes that are currently supported for the three classes of markup:

1. the ***hierarchy***: the structure of the document (chapters, sections, subsections, etc);
2. the block-level structure that is used within the hierarchy: paragraphs, tables, figures, lists, etc (referred to as the ***pool***);
3. the inline markup which occurs in mixed content alongside text within the pool elements, like emphasis, acronyms, GUI objects, links, product names, foreign words, URIs, cross-references, and the names of packages, options, variables, classes, files, etc (referred to as the ***flow***)⁴

Some of the inline element types such as cross-references, bibliographic references, and markup tags have extended features which are described in ‘**Cross-references**’, the third item in the list in [section 6.1 on the next page](#); [section 6.5 on page 70](#) and ‘**<tag>**’, the 38th item in the list in [section 6.4 on page 68](#).

6.1 Writing assistance in XSLT

The following features are designed to help you by automating some common tasks which otherwise cause unnecessary work when editing.

Punctuation: You MAY omit the terminating punctuation in the last (or only) `<para>` in a `<listitem>` in an `<orderedlist>`, `<itemizedlist>`, `<procedure>`, and `<variablelist>`. A semicolon will be used for all items in the list except the last, when a full point (period) will be used, *unless* you have provided punctuation as the final character, or the text node *immediately following* starts with a punctuation character (for example, after an inline list where the sentence continues). See [section 11.4.3 on page 143](#) for the annotated code.

Inline lists: Similarly, you omit the punctuation at the end of each item when using an `<orderedlist>` or `<itemizedlist>` *inside* a `<para>` element, where *inline*, not vertical, formatting is created. These items are *a)* lettered in italics; *b)* fenced with upright parentheses; *c)* punctuated with semicolons;

⁴The terms ‘pool’ and ‘flow’ are taken from the design conventions of DTDs (Maler et al., 1999), whose authors derive them from an Open Software Foundation DTD design committee. They are in widespread use and occur in the specifications for both *DocBook* and HTML, although they appear much earlier under the terms ‘hierarchy’, ‘containment’, and ‘sequence’ in Southall (1989, sec 3).

and *d*) the final item separated with ‘and’ (for an `<orderedlist>`) or ‘or’ (for an `<itemizedlist>`). See [section 11.4.4 on page 162](#) for the annotated code.

Cross-references: The `<xref>` element type with a `@linkend` `<IDREFS>` attribute makes a cross-reference to another element in the document with the matching `@xml:id` attribute value[s]. The `.dtx` file uses the `varioref` package to identify the name of the target object for an extended reference where possible, and the XSLT script provides similar extended references for those objects that `varioref` is unable to identify, like lists.

Supported target element types are:

Whole lists: `<variablelist>`, `<orderedlist>`, `<itemizedlist>`, and `<procedure>`;

List items: `<varlistentry>`, `<listitem>` in `<itemizedlist>` and `<orderedlist>`, and `<step>` in `<procedure>`;

Illustrations: `<table>`, `<figure>`, `<blockquote>`, `<informaltable>`, and `<programlisting>`;

Admonishments: `<note>`, `<warning>`; and `<sidebar>`;

Text: `<para>`.

For a `<listitem>` in an `<itemizedlist>` *ClassPack* will compute the ordinal number of the item. The automated labelling is omitted if `@role` is set to “short”. The page number is included by default: if `@xrefstyle=“page”` then *only* a page reference is given.

You MAY use multiple `<ID>` values in the `@linkend` attribute. See [section 11.4.5 on page 204](#) for the annotated code.

Bibliographic citations: A `<biblioref>` element with a `@linkend` attribute makes a citation of the `<biblientry>` with that `@xml:id[s]` value, which MUST exist in the `<bibliography>`. The default citation format (set for `biblatex` in `prepost.xml`) is author-year in parentheses like this: (Knuth, 1984). If you want to use the author’s name as part of the text of the surrounding sentence, like the book written by Knuth (1984), set the `@xrefstyle` attribute to “text” so that the parentheses enclose only the year. Multiple space-separated values in `@linkend` produce multiple citations like (Flynn, 2009; Holman, 2013; Knuth, 1984). See [section 6.5 on page 70](#) for details, and [section 12.1.1 on page 299](#) for the annotated code.

Identification of terms (for indexing): The following element types are used for the names of components:

- `<classname>` for document classes like `article`;
- `<command>` for commands (macros) like `\maketitle`;
- `<envar>` for environments like `quotation`;
- `<function>` for functions in computer languages other than \TeX ;
- `<option>` for options like `12pt`;
- `<package>` for packages like `geometry`;
- `<parameter>` for parameters like `#3`;
- `<varname>` for variable names like `\textwidth`.

See the full descriptions in [section 6.4 on page 65](#).

6.2 Hierarchical markup

The outline top-level structure is described in [section 3.2 on page 20](#) and [section 3.3 on page 21](#).

The `<part>` elements containing the documentation, annotated code, and any additional files do not have any `<title>` element or direct textual content themselves: they just act as containers to keep them separately identifiable by the XSLT script.

```
<part xml:id="doc" conformance="xelatex">
  <chapter xml:id="ui">
    <title>User interface</title>
    ...
    <sect1>
      <title>Font selection</title>
      ...
    </sect1>
    <sect1 xml:id="margins">
      <title>Margins and spacing</title>
      ...
    </sect1>
  </chapter>
  ...
</part>
```

Within a `<part>`, the REQUIRED top-level division is the `<chapter>`, which equates to a `\section` in the `.dtx` file. Each `<part>` (user documentation, documented code, and additional files) MUST have at least one `<chapter>` element.

The first chapter MAY be preceded by one or more `<preface>` elements, where a preface, foreword, introduction or other preliminary material is needed.

Within chapters, the `<sect1>`, `<sect2>`, `<sect3>` etc elements MUST be used to hold your sections, subsections, and subsubsections. In a `<preface>`, *DocBook* allows only the `<section>`, but this can be nested if required.

You can use as many or as few of these as you feel you need to organise your writing in the documentation part. In the annotated code part, the whole point is to modularise your class or package, so that you can group annotations together to describe each part of the code in a logical and consistent manner.

Each of these hierarchical divisions MUST have a `<title>` element, and MAY also have an `@xml:id` attribute to act as a target (like a `\label` in \LaTeX) for cross-referencing.

In the "doc" part (only), the `<conformance>` attribute SHOULD be set to the name of the \LaTeX processor to be used for producing the documentation. In the example above this is "xelatex"; the default is still "pdf \LaTeX ".

6.3 Structural markup (block-level elements)

Inside your chapters, sections, subsections, etc, you can have any arrangement or mixture of paragraphs, tables, lists, figures, quotations, code samples, and other conventional document components that will be familiar to you from \LaTeX and from other XML environments.

The only requirement in *DocBook* is that each hierarchical division **MUST** contain at least one such structural component. Supported element types at this level are:

- `<annotation>`, see [section 3.6 on page 27](#);
- `<blockquote>`, see [section 6.3.1](#);
- `<bridgehead>`, see [section 6.3.3 on the following page](#);
- `<figure>`, see [section 6.3.5 on page 62](#);
- `<informaltable>`, see [section 6.3.5 on page 62](#);
- `<itemizedlist>`, see [section 6.3.2 on the next page](#);
- `<note>`, see [section 6.3.6 on page 64](#);
- `<orderedlist>`, see [section 6.3.2 on the next page](#);
- `<para>`, see [section 6.3.1](#);
- `<programlisting>`, see [section 6.3.4 on the next page](#);
- `<remark>`, see [section 6.3.7 on page 64](#);
- `<sect1>` / `<sect2>` / `<sect3>`, see [section 6.3.3 on the next page](#);
- `<sidebar>`, see [section 6.3.6 on page 64](#);
- `<subtitle>`, see [section 6.3.1](#);
- `<table>`, see [section 6.3.5 on page 62](#);
- `<task>`, see [section 6.3.7 on page 64](#);
- `<title>`, see [section 6.3.1](#);
- `<variablelist>`, see [section 6.3.2 on the next page](#);
- `<warning>`, see [section 6.3.6 on page 64](#).

The `<annotation>` element is a special case, and is only used in the “code” part (see [section 3.6 on page 27](#)).

6.3.1 Titles, paragraphs, and quotations

`<title>`: a title is **REQUIRED** in all chapters, [sub]sections, figures, and tables; it is **OPTIONAL** in lists, notes, warnings, and other block-level element types.

`<subtitle>`: A `<subtitle>` element can be used for a short-form caption which \LaTeX will use as the optional argument for the Table of Contents;

`<para>`: for normal paragraphs, For tagged content, see [section 6.4 on page 65](#);

`<blockquote>`: The `<blockquote>` element is for quotations from other documents included as `<biblioitem>`s in your `<bibliography>` (see [section 6.5 on page 70](#)), such as useful reference text about why the readers might choose to do things one way in your class or package rather than another. It normally just contains a paragraph or two.

```
<blockquote annotations="in comp.fonts" linkend="pfnut">
  <para>[How do we] persuade new users that spreading
    fonts across the page like peanut butter across
    hot toast is not necessarily the route to
```

```
typographic excellence?</para>
</blockquote>
```

The @xml:id value of the <biblientry> where the document reference is given goes in the @linkend attribute of the <blockquote>.

[How do we] persuade new users that spreading fonts across the page like peanut butter across hot toast is not necessarily the route to typographic excellence?

(Flynn, 1994, in comp.fonts)

As with normal citation, you can give additional details of chapter or page [range], and precite or postcite comments (see ‘<biblioref>’, the second item in the list in [section 6.4 on page 65](#) for details), but the format in block quotations is always (Author, Year).

6.3.2 Lists

<itemizedlist>: for random-order (bulleted) lists — each <listitem> contains one or more <para> elements;

<orderedlist>: for lists where the sequence numbering of items is important; or for lists in which you want numbered items for reference purposes — the structure is the same as for <itemizedlist>;

<variablelist>: for description-style lists like this one — the <term> element in each <varlistentry> holds the reference term; the descriptive part is in the same <listitem> structure as for itemized and ordered lists.

Use the attribute @spacing="compact" on any of <itemizedlist>, <orderedlist>, or <variablelist> for tighter vertical spacing.

If you want to refer to an item in an itemized or ordered list, the @xml:id attribute **MUST** be on the <listitem> element; in variable lists, the @xml:id attribute **MUST** be on the <varlistentry> element.

6.3.3 Subsections

<sect1>, <sect2>, <sect3>: Creates a section inside a chapter. A title element is MANDATORY, plus one further structural element type.

<sect1> may contain one or more <sect2> elements, which in turn may contain <sect3> elements;

<bridgehead>: Creates a cross-head used to break up very long [sub]sections without the need to create a new sub-level.

6.3.4 Code listings

Use the <programlisting> element for listings of code. The @language attribute **MUST** give the computer language except for L^AT_EX (see below).

1. In the "code" part of a *ClassPack* document (the annotated code which is the L^AT_EX code of your class or package), use this element type *without* attributes.

2. In the “doc” part, the basic style is formatted by default (specified in `prepost.xml`) as `\small` type, black text, and `\ttfamily` font. Other attributes can be used to control the appearance:

```
<programlisting language="LaTeXe" wordsize="8/9"
    arch="framed" remap="\itshape" annotations="gdef">
\def\@subtitle{\relax}
\newcommand{\subtitle}[1]{\gdef\@subtitle{#1}}
</programlisting>
```

@language: the language of the code. The default is “LaTeXe”. This is passed to the listings package for formatting, but there are some extras and redefinitions in the `prepost.xml` entry for that package to add to the language support offered;

@wordsize: specifies a different font size, given as the point size and baseline height separated by a slash, eg 8/10. It will also cause the option **breaklines=true** to be passed to the listings package.

If the file or fragment being listed is known to have overlong lines, but font size adjustment is not possible or not wanted, including this attribute with a null value will still invoke line-breaking but leave the font size unchanged.

Note that code listings within the “code” part of a document which are *not* for extraction into the class or package (examples, for example) the Verbatim environment is used for formatting the PDF instead of the macrocode environment. In these cases a font size and baseline size CANNOT be used, and a \LaTeX step-size MUST be used instead (eg `\scriptsize`);

@condition: In code listings within the “code” part of a document which are *not* for extraction into the class or package (examples, for example) this attribute MUST be set to “ignore” in order to prevent it being extracted into the class or package;

@arch: may be set to “framed” to specify a frame drawn round the code;

@annotations: a comma-separated list of additional keywords that you want highlighted in the code (passed to the listings package);

@remap: a \LaTeX command or sequence of commands to modify the highlighting of the keywords given in the @annotations attribute.

3. For documenting a long file in the annotated code, it is easier to do it chunk by chunk, including the chunks (fragments) directly from the file itself by start/end line number, instead of retyping them or copying and pasting.

This is done by giving the file name in the `@xlink:href` attribute with the `@startinglinenumber` and `@endinglinenumber` set to the line-numbers of the lines to start and end the chunk (inclusive). In this mode, the `<programlisting>` element is EMPTY (no content).

@xlink:href: name of the file from which to take the code (and path, if the file is not in the same directory as the document);

@startinglinenumber: a line number to start a fragmentary listing at; *or* a label (see below);

@endinglinenumber: a line number to stop a fragmentary listing at; *or* a label (see below).

However, as start/end line-number references mean the numbers have to be updated every time the file is edited, the two attributes `@startinglinenumber` and `@endinglinenumber` can instead be set to labels (one-word strings), and the labels **MUST** be inserted in the file as comments, shown in the table below *a)* on a line by themselves; *b)* starting in character position 1; *c)* with a single space character where shown; and *d)* label values must be unique within the file.

Language	Label format	Example
\LaTeX	Double comment character	<code>%% label %%</code>
<i>bash</i> (1) script	Double comment character	<code>## label ##</code>
XML and XSLT	Processing Instruction (PI)	<code><?cpdoc label?></code>
CSS	Start- and end-comment	<code>/* label */</code>

If the code being reproduced has been pre-chunked into separate files (see item 3 below and the note in 3), set the `@conformance` attribute to "frag" (currently the only value supported).

Listing bug in chunking

The listings package currently (v.1.6 of 2015/06/04) has a bug in interpreting labels in XML documents under certain circumstances. To overcome this, the utility script *chunk* provided with *ClassPack* can be used to split the file into separate chunks. In this case, the labels at the start and end of a chunk **MUST** be the same (ie they must be in pairs). Each chunk file gets named after the original file plus the name of the label (eg foobar-intro), and the extension .frag, which **MUST** be specified in the `@conformance` attribute. In this circumstance, *only* the `@startinglinenumber` need be used, set to the label which has become part of the fragment's filename; there is no need for the `@endinglinenumber`

6.3.5 Figures and tables

<figure>: for Figures, using the structure shown:

```
<figure floatstyle="hb" xml:id="me">
  <title>A picture</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="mypic.png" width="90%"/>
    </imageobject>
  </mediaobject>
</figure>
```

```
</figure>
```

The `@floatstyle` attribute is used to specify the float parameter option for the \LaTeX figure environment; the image filename **MUST** go in the `@fileref` attribute of an `<imagedata>` element in an `<imageobject>` element in a `<mediaobject>` element; and the `@width` attribute **MAY** be used to give the width as a percentage (as shown) or in any acceptable \LaTeX unit (default is full width of the text).

<table>: for formal Tables, using the structure:

```
<table floatstyle="p" xml:id="costs">
  <title>Example</title>
  <tgroup cols="3">
    <colspec align="left"/>
    <colspec align="center"/>
    <colspec align="right"/>
    ...
```

In addition to the column alignments shown, other attributes can be used on the `<colspec>` element type to control the formation of the \LaTeX column specification:

@condition: \LaTeX commands to be executed at the start of each cell in this column;

@colwidth: specifies the width of the column as a percentage (of available width) or in any acceptable \LaTeX unit;

@char: the vertical alignment of a paragraphic (multi-line) column; the default is `p` (top-aligned) but may be `m` (middle-aligned) or `b` (bottom-aligned) for columns with a width specified, as implemented in the array package. The `d` value for decimal alignment is only partially implemented in this version;

@conformance: \LaTeX commands to be executed at the end of each cell in this column.

```
<thead>
  <row>
    <entry>Item</entry>
    <entry>Code</entry>
    <entry>Cost</entry>
  </row>
</thead>
<tfoot>
  <row>
    <entry>Total</entry>
    <entry></entry>
    <entry>100</entry>
  </row>
</tfoot>
...
```

Note that the `<tfoot>` **MUST** come before the `<tbody>`.

```

<tbody>
  <row>
    <entry>&LaTeX;</entry>
    <entry>LA</entry>
    <entry>Ø</entry>
  </row>
  <row>
    <entry>Word</entry>
    <entry>WD</entry>
    <entry>1ØØ</entry>
  </row>
</tbody>
</tgroup>
</table>

```

Columns can be spanned by using the @spanname attribute to give the number of columns to span (the unused cells **MUST** still be included, empty). There is no provision in this version for row spanning.

The @align, <colwidth>, and @wordsize attributes MAY also be used on the <entry> element to change the alignment, column width, and font size for individual cells, as with the <colspec> element. The @wordsize works the same way as for code listings, see ‘@wordsize’, the second item in the list in 2 on page 61.

A <row> element MAY have a @role of “header” to specify that a row is to be used as a crosshead (set in bold type by default).

If the @rowsep attribute on a <row> element is set to “1”, the row will be given 1 ex of vertical separation *from the preceding row* (no other value is possible in this version).

<informaltable>: A tabular array without a caption or number, and which does not float, is done with the <informaltable> element. This contains a <tgroup> which is structured identically to the formal table described above.

6.3.6 Admonishments

<note>: for note blocks (framed and centred) giving special or unusual information about the current topic;

<sidebar>: for sidebars (floating panels neither figures nor tables, usually holding ancillary explanatory text); a <title> is recommended;

<warning>: for warning blocks (framed and centred).

6.3.7 Tasks to do, and remarks when done

<task>: This element is used to mark note-to-self TODO tasks in your code or documentation. They display in white on black so they stand out.

```

<task>
  <procedure>
    <step>
      <para>The <package>babelfish</package>
        seems to be redundant here now</para>
    </step>
  </procedure>
</task>

```

```

    </step>
  </procedure>
</task>

```

The `<task>` element MUST contain a `<procedure>` subitem containing at least one `<step>`, with the text in a `<para>`;

<remark>: This element is used for notes and comments about changes and bug fixes to your code which you want included in the change log.

```

<remark version="3.44" revision="2021-04-01">Fixed bug
where the counter was not being checked for range</remark>

```

The `@version` and `@revision` attributes MUST give the version number and the date, respectively.

6.4 Inline markup (element types in mixed content)

Element types marked with a star (*) are intended for use primarily in annotations, as they generate special `dox` package commands defined in `prepost.xml` for adding \LaTeX (and XML) terms to the index, and highlighting them in the left-hand margin to flag their argument. To annul this when used elsewhere, set the `@condition` attribute to `"nodesc"` on these starred elements.

<acronym>: Identifies an acronym and formats it in a small-caps font. To set the defining instance, give the acronym itself in the `@xml:id` attribute and give the text expansion as the element content. Otherwise just give the acronym as the content. If a defining instance for an acronym exists elsewhere in the document, the content becomes a link to it.

<biblioref>: a citation of an item in the Bibliography; the `@linkend` attribute MUST be the value of the `@xml:id` of the `<biblioentry>` element to which it refers (see [section 6.5 on page 70](#)). Multiple entries can be cited: separate the values in `@linkend` with white-space.

The citation style can be varied by setting the `@xrefstyle` attribute to one of: `"text"` [the Author (Year) style] or `"paren"` [the (Author, Year) style, the default]; or one of `"full"`, `"title"`, `"shorttitle"`, `"author"`, or `"shortauthor"`, for citing those fields only; or `"field"` with the `@remap` attribute set to the \LaTeX name of the field required.

If the citation is of a specific location in the document referenced, set the `@units` attribute to `"ch"` (chapter), `"p"` (page) or `"§"` (section), and set the `@begin` attribute to the number of the unit (and optionally set the `@end` attribute for a range).

The `@role` and `@annotations` attributes are for additional comments that you want included (respectively) before or after the citation itself.

Note that this usage applies also to the `<blockquote>` element when used for quotations from a cited source (see `<blockquote>`, the last item in the list in [section 6.3.1 on page 59](#)).

<citetitle>: the title of a document being mentioned, without being a formal citation; by default this is formatted in italics. It may be empty, with a @linkend attribute pointing to the <xml:id> of an entry in the Bibliography, in which case the title produced the same way as for <biblioref> with an @xrefstyle set to "title" (in which case it is italicised for monographs and in quotes for articles).

* **<classname>**: a L^AT_EX document class name like article;

<code>: a fragment of computer or data code, set in monospace type;

* **<command>**: a L^AT_EX or other computer command, such as \parskip. In the (default) case of L^AT_EX, do NOT give the backslash, as it is added automatically; other languages REQUIRE the @language attribute giving the name of the language (eg "bash", "XML", "XSLT", etc);

<coref>: [ab]used as a function to return the number of items in a list. The @xml:id of the target list MUST be given in the @linkend attribute, and the name of the element type to count MUST be given in the @xreflabel attribute;

<emphasis>: emphasis according to style, usually italics;

* **<envvar>**: a L^AT_EX environment name like enumerate;

<exceptionname>: used for the keywords of RFC2119 in formal admonishments;

<filename>: name of a file, a full filepath, or just a part of the name (eg a filetype);

<firstterm>: the defining instance of a specialist term; it may or may not actually be the first occurrence;

firstterm

TODO: Make this work like acronyms

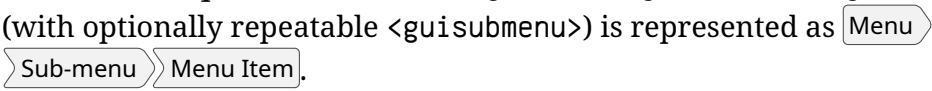
<footnote>: a footnote;

<foreignphrase>: for foreign-language expressions; identify the language with the two-letter ISO 639-1 code in the @xml:lang attribute if the phrase is long enough to need the babel package (loaded automatically);

* **<function>**: identifies a function name in a programming language named in the @language attribute;

<guibutton>: represents a GUI  (round corners);


<guilabel>: represents a GUI  (square corners);

GUI menus: the sequence of elements <guimenu> <guisubmenu> <guimenuitem> (with optionally repeatable <guisubmenu>) is represented as .

<inlineequation>: L^AT_EX code for a mathematical expression to be rendered inline;

<inlinemediaobject>: include a text-sized image, for example an icon unavailable by other means. This **REQUIRES** the `<imageobject>/<imagedata>` subelements just like the images in `<figure>` do;

<itemizedlist>: an *inline* list with ‘or’ before the final item, formatted within the paragraph, with italic lowercase letters and a parenthesis denoting the items. The separating semicolon and terminating ‘or’ are automatic (courtesy of the `enumitem` package) and **MUST NOT** be typed;

<keycap>: portrays a specific key on the keyboard such as  when other means are inadequate;

<link>: an external reference to a URI using the `@xlink:href` attribute. If the element has content, it is formatted as a link; if the element is empty, the URI is shown as the link text;

<literal>/<literallayout>: identifies a literal string of \LaTeX code on which no interpretation is to be performed in XML (markup characters like backslashes and curly braces will not be escaped but passed through to the `.dtx` file untouched); the `@xml:lang` attribute can be set to the name of the language (eg `"TeX"`, `"LaTeX"`, etc);

<markup>: identifies markup in languages *other than* XML or \LaTeX . Use the `@language` attribute to give the language;

<methodname>: identifies a template or method in a programming language (eg XSLT). Use the `@language` attribute to give the language;

olink: used as a pointer in the `readme.xml`: see [section 3.8 on page 31](#);

* **<option>**: a \LaTeX option to a class, package, or command, like **a4paper**;

<orderedlist>: an *inline* list with ‘and’ before the final item, formatted within the paragraph, with italic lowercase letters and a parenthesis denoting the items. The separating semicolon and terminating ‘and’ are automatic (courtesy of the `enumitem` package) and **MUST NOT** be typed;

* **<package>**: a \LaTeX package name like `fancybox`;

* **<parameter>**: used in annotating XSLT parameters; behaves as `@varname="parameter"`;

personname: this is for highlighting the names of people, usually contributors or authors of other classes or packages. It **MUST** contain their `<firstname>` and `<surname>` and it will create an index entry for them as well as being in the text;

<phrase>: marks a phrase used as-is (ie not a quote from anyone in particular) by enclosing it in quotation marks;

<productname>: marks a product name in italics. Can also be used for system commands, eg *make* (1). The `@userlevel` attribute can be set to the chapter number of the UNIX manual pages for the command;

<quote>: marks a quote from someone by putting it in quotation marks;

<replaceable>: identifies text, commands, or keywords to be typed by the user in an example, *for which the user must substitute a meaningful value*, eg password (shown in italics when within a monospace example, or underlined if in normal text);

<systemitem>: identifies generic computer-related strings such as system commands, hostnames, Regular Expressions, etc which need to be printed in monospace to eliminate any confusion over 1/l/I, 0/O, etc;

<tag>: an XML element, attribute, attribute value, or entity name: the type is specified in the @class attribute and is one of the predetermined list provided automatically by *DocBook* (your XML editor will guide you).

To suppress the automated ‘at’ (@) prefix on attributes, set conformance="noat".

<type>: marks a span of text for which specific typographical treatment is needed, such as examples of output. The following attributes control the appearance:

@fontencoding: one of the values T1, OT1, or U. The default is not to change the current encoding setting;

@fontfamily: one of "roman", "sans", or "monospace" for selecting that family from among the three always available in L^AT_EX.

If this is the only attribute, and the element is empty, it will cause the name of the current font family to be output;

@fontface: the ‘Karl Berry’ font family name of an installed font such as ptm (Times), bch (Charter), etc;

@fontshape: one of "it" (italic) or "up" (upright) or "sc" (small caps).

@fontseries: one of "k" (book), "r" (regular), "m" (medium), "n" (normal), "b" (bold), "bx" (extra bold), "it" (italic), "sl" (slanted), "l" (light), or "c" (heavy). Others may be used if you know that your font setup supports them;

@fontsize: *either* the font size in the format `pp/bb` for the point size and the baseline height (same as for ‘@wordsize’, the second item in the list in 2 on page 61); *or* just a point size, in which case the baseline height will be set to zero.

Note that if @fontsize is used with the slash syntax, the content of the element will be terminated by a paragraph break in order to honour the baseline height.

@fontcolor: a colour name from an installed palette of the xcolor package, default `svgnames`.

For example, *CornflowerBlue, bold, italic, sans-serif, 16pt*

```
<type fontfamily="sans" fontcolor="CornflowerBlue"
fontseries="b" fontshape="it">
```

```
fontSize="16">CornflowerBlue, bold, italic,
sans-serif, 16pt</type>
```

These are experimental and may change slightly in future versions. The terminology (except `@fontcolor`) is standard \LaTeX .

<uri>: (deprecated) marks a URI (formats it with the `\url` macro). This is being replaced by the `<link>` element type (see '`<link>`', the 23rd item in this list).

- * **<varname>**: a \LaTeX variable, like a length or a counter, or a variable in another language specified in the `@language` attribute. The `<role>` attribute is used to specify the type of variable: for \LaTeX the value is usually "length" or "counter", but there is a special case of "color" (note spelling, and see [section 3.6.1.1 on page 29](#)). The backslash on length names **MUST NOT** be typed (same rule as with macro names in the `<command>` element above) as it will be added automatically so that the name can be indexed properly. For \LaTeX field names, use the `@role` value "field". Use the `@language` attribute for "BiBTeX", "bash", "XML", "XSLT", etc.

When used in annotations, the name of the variable is echoed unheralded in the margin. If the element is used in the first line of an annotation, this echo may overtype the name of the annotation; *ClassPack* tries to avoid this but the echo can be suppressed manually by setting the ;

<wordasword>: marks a word that is being used as itself (usually for purposes of clarification), so it goes in quotation marks;

<xref>: a cross-reference using the `@linkend` attribute to point to one or more elements with an `@xml:id` attribute somewhere else in the document; the mechanism is identical to \LaTeX 's `\label...\ref` but the presentation uses (mainly) the extended labelling features of the `varioref` package to enable the labelling of cross-references with the name of the target object. A cross-reference to a Figure, for example, is automatically labelled 'Figure...' — you do not (indeed **MUST NOT**) type the name of the target.

Some target element types are not recognised by `varioref` and are therefore compensated for in the XSLT script (see [section 11.4.5 on page 204](#)).

Note that these automations are a work-in-progress. In particular they are not yet consistent, and insufficiently context-sensitive.

<varlistentry>: formats the term in bold, in quotes, and calculates the ordinal position in the list as a number if over 12 or as a word if under; and then uses `varioref` to provide the section and page reference; eg the entry for the `<type>` element earlier would read: '`<type>`', the 39th item in the list in [section 6.4 on the preceding page](#)

<listitem> in <itemizedlist>: calculates the ordinal position in the same way as for `<varlistentry>` above; and then uses `varioref` to provide the section and page reference; eg the entry for the `.ins` file at the start of

this document would read: the second item in the list in [section 2 on page 13](#)

<listitem> in <orderedlist>: uses the item counter and the page reference with a standard `\ref`; eg the entry explaining ‘flow’ elements would read: item [3](#) in the list on p. [56](#)

<step> in <procedure>: works the same way as `<listitem>` in `<orderedlist>`.

List references should add ‘above’ or ‘below’ if the target is in the same list, but this has not been evenly implemented yet.

<orderedlist>: for a whole ordered list with a title, the reference quotes the title and gives a section reference using `varioref`;

<informaltable>: refers to ‘the table’ and uses `varioref` for the section and page;

<programlisting>: refers to ‘the code’ and uses `varioref` for the page;

<note>, <warning>, <sidebar>: refers to the name of the element and uses `varioref` for the page;

<part>: references to a location inside an extractable file in the ‘files’ part experimentally creates a hyperlink to the target;

all others: use `\vref`.

There are some exceptions:

@xrefstyle="page": this attribute value forces a page reference only (the default for `<informaltable>` or `<programlisting>`) using `\vpageref`;

Ranges: if the target is *not* `<orderedlist/listitem>` *and not* a `<procedure/step>` *and not* a location in the “files” `<part>`, and if the `@endterm` is present, use `\vrage` on `@linkend` and `@endterm`;

@xreflabel: if there is an `@xreflabel` attribute present, put the content of the target’s first subelement named in that `@xreflabel` in (parentheses).

6.5 Bibliography

If you use bibliographic citation and reference in your documentation, the references themselves **MUST** be stored in a `<bibliography>` element immediately after the last `<chapter>` element in the “doc” part. This **MUST** contain a `<biblioentry>` element for each reference you wish to cite (for how to make a citation, see ‘`<biblioref>`’, the second item in the list in [section 6.4 on page 65](#)).

This enables your XML editor to use the `<ID>/<IDREF>` links between `<biblioref>` and `<biblioentry>` for validation, and to ensure you don’t cite something not in your references, and that only references that are cited get output.

6.5.1 The <bibliography> element

The attributes on the <bibliography> start-tag identify the type of bibliographic formatting required:

```
<bibliography arch="biblatex" condition="biber" label="apa"
  xlink:href="foo.bib">
  <biblioentry...>
    ...
  </biblioentry>
  ...
</bibliography>
```

@arch: specifies which type of handler is to be used, either "biblatex" for the biblatex package, or "bibtex" for the *bibtex* program. The default is "biblatex";

@condition: specifies the bibliographic processor (program) to use, either "biber" or "bibtex"; this is only meaningful in the case of arch="biblatex", and is otherwise ignored. The default is "biber";

@label: specifies the bibliographic style of reference. This **MUST** be the name of an installed style recognised by *biblatex* or *bibtex*, whichever was specified in @arch above. The default is "apa" for *biblatex* and "apacite" for *BIB_TEX*;

label

TODO: Maybe add all valid style names to the DTD

@xlink:href: OPTIONAL gives the name of the output *BIB_TEX* file (*including* the .bib filetype) to which the references should be written by the XSLT script. The default is the same as the name of the class or package being created. This attribute would typically be used only where the package or class being documented is part of a suite.

@xreflabel: gives the name of any required ancillary package for the style specified in the @label attribute above (often called by the same name, eg apacite, natbib, chicago, etc). Note that this is currently used *only* when using *BIB_TEX*, *not biblatex*.

xreflabel

TODO: Add all possible names to the DTD?

If the <bibliography> element immediately contains a <title> element, the value is used as the \bibname for the heading on the bibliography.

6.5.2 Bibliographic entries

Every <biblioentry> element **MUST** have *both* an @xml:id attribute by which it can be cited with the <biblioref> element *and* an @xreflabel attribute classifying it with one of the standard *BIB_TEX* or *biblatex* document types (article, book, incollection, etc, as specified in the biblatex documentation). Currently-supported values can be found in the *DTD* and will be displayed by your XML editor.

The key to successful citation and reference is to ensure that documents are accurately described. The most common types are shown in [Table 1](#).

Table 1: Common $\text{BIB}_{\text{T}}\text{X}$ entry types

Entry type	Used for	Analytic relation	Monographic relation
article	Article in a journal	"article"	"journal"
book	Books		
inbook	Chapter in a book all written by the same author	"chapter"	"book"
incollection	Article in a book all written by different people	"article"	"book"
inproceedings	Paper given at a conference and published in Proceedings	"paper"	"proceedings"
manual	Documentation, especially for another class or package		
online	Web page or downloaded document		
report	Technical report		
standard	International or national standards		
thesis	Doctor's or Master's dissertation or thesis		

Care is needed *in particular* for those entries for documents contained within other documents, eg article in journal, chapter in book, article in collection, paper at conference, etc. Use the `<biblioset>` element to keep the two parts of such entry separate as in this example:

```
<biblioentry xml:id="tb97" xreflabel="article">
  <biblioset relation="article">
    <author>
      <personname>
        <surname>Flynn</surname>
        <firstname>Peter</firstname>
      </personname>
    </author>
    <title>'Typographers Inn</title>
    <subtitle>Where have all the flowers gone?</subtitle>
    <artpagenums>21-22</artpagenums>
  </biblioset>
  <biblioset relation="journal">
    <title>TUGboat</title>
    <volumenum>31</volumenum>
    <issuenum>1</issuenum>
    <date YYYY-MM-DD="2010"/>
  </biblioset>
</biblioentry>
```

The bibliographic XSLT transformation to $\text{BIB}_{\text{T}}\text{X}$ is encoded in `db2bibtex.xsl`, annotated in [section 12 on page 299](#).

Items with containers: Articles in journals, articles (by different authors) in collections, chapters in books, and papers at conferences all require two `<biblioset>` elements as in the example above: one ('analytic') to hold the

title and author of the article, chapter, or paper, and its page range; and one (‘monographic’) to hold the details of the publication or conference. The value of the @relation attribute for the two <biblioset> elements is given in [Table 1 on the previous page](#).

Items without containers: All other types of reference, eg books, manuals, reports, standards, whole issues of periodicals or proceedings, theses, and standalone online resources, etc do not require a <biblioset> element.

6.5.2.1 Title and subtitle : Use the <title> element for the title of a document. If there is a subtitle (often separated from the title by an em-rule or a colon) put that in a <subtitle> element.

In the items shown in [Table 1 on the preceding page](#), there will normally be a <title> in *both* <biblioset> elements: one for the title of the article, chapter, or paper; and one for the title of the journal or book (for conferences there is a separate <conftitle> element).

If the title is long, the <titleabbrev> element MAY be used to give a shortened version of the title.

6.5.2.2 Authors and editors : Single-author documents just use the <author> element. For a named person, use the <personname> sub-element to hold the <firstname>, <othername> (optional), and <surname> elements. For a corporate author (company name), use the <orgname> instead of <personname>; if there is a division or office name, put it in an <orgdiv> element.

For multi-authored documents you MUST enclose *all* the <author> elements in an <authorgroup> container element.

An <affiliation> element MAY be used *after* the <personname> element where it is important to identify it.

The same rules apply to editors (usually only for conference proceedings and books of articles by different authors). These go in the monographic <biblioset>. For multiple editors, enclose them in an <authorgroup> element — there is no <editorgroup> element in *DocBook*!

6.5.2.3 Publication metadata : There will in all cases be a <title> element in a monographic <biblioset>, as explained above (and maybe a <subtitle> and a <titleabbrev>). There may also be editors, as explained in [section 6.5.2.2](#). Not all the other metadata listed here will apply to all entries.

Dates: Use the <date> element but put the date into the @YYYY-MM-DD attribute in that format (ISO 8601 standard date with hyphens separating the two-digit month and day). DO NOT use the L^AT_EX form separating the month and the day with slashes. There SHOULD NOT be any text content in a <date> element (it will be ignored). If an exact date is not available, use the year and give the month if known, or 12 for December and 31 for the day: it MUST still be a valid ISO 8601 date.

The @YYYY-MM-DD attribute is an addition in *ClassPack* as described in item 2 in the list on p. 16, and is not standard *DocBook*.

Publisher: Use the <publisher> element with the sub-elements <publishername> for the name and <address> for the location. The address is normally just the city (and state or country if needed).

Other organisations: Where there is no formal publisher, but the document is issued by or obtainable from an institution or organisation (as for example with a thesis or report), use the <org> element with the sub-elements <orgname> for the name, <orgdiv> for the division or section, and <address> for the location (same rule as for Publisher above).

This is also suitable for the informal publication of web pages such as personal blogs, giving the hosting provider if the site is not the owner's own host.

Edition: Use the <edition> element and give the cardinal number (eg 1, 2, 3), *not* the ordinal (1st, 2nd, 3rd).

Volume and issue numbers: Use <volumenum> and <issuenum> for the volume and issue of journal articles.

Where an item has a number that is not a volume or issue, such as with a report, use the <biblioid> element described in 'Identity', the last item in this list.

Page numbers: For the page number or range of pages for an article or chapter, use <artpagenums> in the analytic <biblioset>.

For the page count (total number of pages in a book), use <pagenums> in the monographic <biblioset>.

Identity: For official numbering schemes (eg ISBN, ISSN, DOI, etc), and for URIs and in-house numbering schemes, use the <biblioid> element and set the @class attribute appropriately.

If there is no appropriate value for the @class attribute, use "other" and set the <otherclass> to a suitable value of your own.

For online resources (eg "uri"), set the @YYYY-MM-DD attribute to the date when you retrieved or downloaded the page or document.

6.5.2.4 Conference metadata : Conference information is slightly different from other types of publication. It **MUST** all be contained within a <confgroup> element.

Conference title (name): Conference titles may range from simple abbreviations or single words (often including the year) up to entire sentences several lines long. In *Classpack* processing, the BibTeX field value is prefixed automatically with 'In Proc.' (proceedings), so you **MUST NOT** include those words in the title.

If the `<confnum>` element is present (see below), any year value at the start or end of the title (eg "2014 EPDIC" or "TUG'95") is removed by *ClassPack* and replaced with the full year or sequence value from the `<confnum>` element.

Conference sponsor: A sponsor name in the `<confsponsor>` element is assumed to be the hosting organisation, and the value is passed to the organisation BibTeX field.

Conference number: The `<confnum>` element MAY contain either a year or a sequence value;

Conference dates: The date attributes for the `<confdates>` element are *different* from the normal `@YYYY-MM-DD` found on other dates. Instead there are two: `@YYYY-MM-DD-from` and `@YYYY-MM-DD-to` for the begin and end dates. Both MUST be ISO 8601 dates. The element is EMPTY (any content is ignored).

Conference location: Use the `<address>` element.

7 Producing your class or package

At the simplest level, the `.dtx` and `.ins` files (and any ancillary files specified) can be generated by running the `db2dtx.xsl` script on your *ClassPack* XML document, using Java and Saxon with a command such as the one given below, substituting the name of your package or class file and the paths to your package or class and to *ClassPack*'s XSLT program and files. It could equally well be done using the built-in API in many XML editors such as *oXygen*.

```
java -Djdk.xml.entityExpansionLimit=0 \
-jar saxon9he.jar \
-xi:on \
-s:mypackage.xml \
-o:mypackage.dtx \
-dtd:on \
-xsl:db2dtx.xsl \
appdir=/path/to/mypackage \
cpdir=/path/to/classpack
```

As mentioned in the note in [section 3 on page 18](#), you MUST keep each class or package you manage in separate directories, named after the class or package, and that name must also be the `@xml:id` value of the root element of the document.

7.1 Maintaining your class or package

The things to check each time you make an update are:

1. update the `@version` and `@revision` attributes on the `<book>` root element;
2. add a new `<revision>` element in the `<revhistory>`, setting the `@version` attribute to the compound of the version and revision specified above, and setting the `<date>` subelement's `@YYYY-MM-DD` attribute to today's date in ISO format;
3. after processing the document once, set the `<book>` root element's `@security` attribute to the checksum displayed by \LaTeX ;
4. process the document again and the security checksum should now match.

8 References

- Bradner, S. (1997). *Key words for use in RFCs to Indicate Requirement Levels* (tech. rep.). Internet Engineering Task Force. Reston, VA, Internet Standard. <https://tools.ietf.org/rfc/rfc2119.txt>
- Even, B. (Ed.). (2002). *ChkTeX: L^AT_EX SEMANTIC CHECKER*. Nesher, Israel.
- Flynn, P. (1994). *Re: Pixel widths in em square* [In Usenet newsgroup comp.fonts, also available from Google Groups]. Retrieved July 10, 2020, from <https://groups.google.com/d/msg/comp.fonts/VYb8QF-2o3U/dGWS6LIwFF4J>
- Flynn, P. (1999). The vulcan package: a repair patch for L^AT_EX, In *Proc. TUG 1999*, T_EX Users Group. Vancouver, BC. <https://www.tug.org/TUGboat/tb20-3/tb64flynn.pdf>
- Flynn, P. (2009). Why writers don't use XML: The usability of editing software for structured documents, In *Proc. Balisage—The Markup Conference 2009*, Balisage Series on Markup Technologies. Montréal, QC. <http://www.balisage.net/Proceedings/vol3/html/Flynn01/BalisageVol3-Flynn01.html>
- Flynn, P. (2013). Markup to generate markup to generate markup: Using XML to create and maintain L^AT_EX packages and classes, In *Proc. Balisage—The Markup Conference 2013*, Balisage Series on Markup Technologies. Montréal, QC. <http://www.balisage.net/Proceedings/vol10/html/Flynn01/BalisageVol10-Flynn01.html>
- Flynn, P. (2014). *Human interfaces to structured documents* (Doctoral dissertation). University College Cork. <https://cora.ucc.ie/handle/10468/1690>
- Grothmann, R. (1993). *check: a T_EX syntax checker and tidier*. Portland, OR.
- Hagen, H. (2001). *ConT_EXt the manual*. Pragma-ADE. 8061GH Hasselt, NL. Retrieved March 27, 2013, from <http://www.pragma-ade.com/general/manuals/cont-eni.pdf>
- Holman, K. (2013). Re: [xml-dev] Excellent quote from Len Bullard. *OASIS XML Developers' Mailing List Archives*, 201310(57). <http://lists.xml.org/archives/xml-dev/201310/msg00057.html>
- Kay, M. (2013). Re: [the XML Guild] Implied semantics of element type names [Author's archive]. *XML Guild mailing list*.
- Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, 27(2).
- Maler, E. & el Andaloussi, J. (1999). *Developing SGML DTDs: from Text to Model to Markup*. Upper Saddle River, NJ, Prentice-Hall.
- Mittelbach, F. (2016). *The doc and shortvrb Packages*. Portland, OR.
- Pakin, S. (2015). *How to Package Your L^AT_EX Package*. T_EX Users Group. Portland, OR.
- Perin, F., Renggli, L. & Ressia, J. (2013). *TextLint*. Bern, Switzerland.

- Southall, R. (1989). Interfaces between the Designer and the Document. In J. André, R. Furuta & V. Quint (Eds.), *Structured Documents*. Cambridge, England, CUP.
- Spring, N. (2011). *style-check*.
- Thorup, K. K. & Abrahamsen, P. (1997). *lacheck*. Portland, OR.
- Walsh, N. (2011). *DocBook 5: The Definitive Guide*. Sebastopol, CA, O'Reilly.
- WorldWideWeb Consortium XSLT Working Group. (2017). *XSL Transformations (XSLT) Version 3.0* (M. Kay, Ed.). Boston, MA, W3C. <https://www.w3.org/TR/xslt-30/>

Code for the Package code for supporting *ClassPack* documentation

9 The classpack macros and settings

9.1 Auto-initialisation

This section is added automatically by *ClassPack* as a preamble to all classes and style packages. For details see the ltxdoc package documentation.

```
1 \NeedsTeXFormat{LaTeX2e}[2015/01/01]
2 \ProvidesPackage{classpack}[2024/02/21 v1.28
3   Package code for supporting ClassPack documentation]
```

9.2 Packages required for the classpack package itself (used *only* when producing documentation for packages authored using *ClassPack*)

Packages required for operation:

noto Sets the Google NoTo typeface as the default.

```
4 \RequirePackage{noto}%
```

fancyhdr Provide for running headers and footers.

```
5 \RequirePackage{fancyhdr}%
```

ltxcmds Some \LaTeX kernel commands for general use, but in the case of *ClassPack*, particularly `\ltx@ifpackageloaded`.

```
6 \RequirePackage{ltxcmds}%
```

Define the `\ltx@ifpackageloaded` command.

```
7 \ifdefined\IfPackageLoaded\relax
8   \else\newcommand{\IfPackageLoaded}[3]{%
9     \ltx@ifpackageloaded{#1}{#2}{#3}}\fi
```

parskip Creates paragraphs separated by white-space with no indentation.

```
10 \RequirePackage{parskip}%
```

graphicx Provide for graphics (PNG, JPG, or PDF format (only) for pdf_latex; EPS format (only) for standard \LaTeX); and for reflection and rotation features.

```
11 \RequirePackage{graphicx}%
```

marginnote Adds more flexibility to marginal notes.

```
12 \RequirePackage[fulladjust,quiet]{marginnote}%
```

9.3 Index settings

`IndexColumns` The doctex package uses a default three-column index for the documentation, which is too narrow for most purposes. We therefore make the index in two columns, and space them slightly farther apart. We test first for the existence of the counter, in case this gets used in a document other than a .dtx file. No such test is needed for `\columnsep` because it is defined in the \TeX kernel.

```
13 \ifundefined{c@IndexColumns}{}{\setcounter{IndexColumns}{2}}
14 \setlength{\columnsep}{3pc}
```

9.4 Annotation settings

`\MacroFont` The doc and docx packages use the `\MacroFont` command for the annotated code. We redefine it here to add the colour DarkBlue (from the `svgnames` option to the `xcolor` package).

```
15 \def\MacroFont{%\fontencoding\encodingdefault
16   \ttfamily\fontseries{m}\fontshape\updefault
17   \small\selectfont\color{DarkBlue}}
```

`\CPKrunningecho` This allows alignment of the current annotation name (from `@xreflabel`) as a reminder in a marginal note in a fake subheading implemented by a `<bridgehead>` element. Again, no at-sign for a user-mode command.

```
18 \newcommand{\CPKrunningecho}[1]{\leavevmode
19   \marginnote[\sloppy\raggedleft\color{Grey}%
20     \hspace{0pt}\LabelFont{#1}]{%
21     {\sloppy\raggedright\color{Grey}%
22       \hspace{0pt}\LabelFont{#1}}}%
23 }
24 \let\marginfont\ttfamily
```

9.5 Table of Contents

`\l@section` Documentation can sometimes have more than nine subdivisions in sections, subsections, etc, and over 99 pages; and the default widths in the ToC are too narrow for this, so we widen the space for the subsection number by 0.4em:

```
25 \renewcommand*\l@section{%
26   \@dottedtocline{2}{1.5em}{2.7em}}
```

`\l@subsubsection` Similarly we increase the subsection number space by 0.4em, and its margin, so they align:

```
27 \renewcommand*\l@subsubsection{%
28   \@dottedtocline{3}{4.2em}{3.6em}}
```

`\@pnumwidth` The page number width is set to 3em instead of 1.55em:

```
29 \renewcommand{\@pnumwidth}{3em}
```

`\@tocrmarg` And the right margin space goes up from 2.55em to 3em; the addition of 1fil makes the section titles typeset raggedright, so that hyphenation will not occur.

```
30 \renewcommand{\@tocrmarg}{4em plus1fil}
```

9.6 Lower-level sectioning

`\subsubsection` The `\subsubsection` command is used in bridgehead mode, so needs less space above and below.

```
31 \renewcommand\subsubsection{%
32   \@startsection{subsubsection}{3}{\z@}%
33   {-1ex\@plus -.25ex \@minus -.25ex}%
34   {1ex \@plus .25ex}%
35   {\sffamily\normalsize\bfseries}}
```

9.7 Appendix settings

`\appendix` Change the way the appendix command works so that appendixes get section-type styling in documentation.

```
36 \renewcommand\appendix{\par
37   \setcounter{section}{0}%
38   \setcounter{subsection}{0}%
39   \gdef\thesection{\@Alph\c@section}}
```

9.8 T_EX and other logos

T_EX and L^AT_EX are defined in the L^AT_EX kernel, but most of the others are not. The following definitions are taken from the ltugboat package, used for typesetting the TUGboat journal.

`\ConTeXt` ConT_EXt is a typography and typesetting system meant to provide users easy and consistent access to advanced typographical control (Hagen, 2001).

```
40 \def\ConTeXt{C\kern-.0333emon\-\kern-.0667em\TeX
41   \kern-.0333emt}
```

`\tubreflect` Borrow the reflection code from TUGboat.

```
42 \def\tubreflect#1{%
43   \@ifundefined{reflectbox}{%
44     \PackageError{classpack}%
45     {A graphics package must be loaded for \string\XeTeX}%
46     {Add the graphicx package to your Preamble}%
47   }{% otherwise OK
48     \ifdim \fontdimen1\font>0pt
49       \raise 1.75ex \hbox{\kern.1em
50         \rotatebox{180}{#1}}\kern-.1em
51     \else
52       \reflectbox{#1}%
53     \fi}
```

```

53     \fi
54   }%
55 }

```

`\tubhideheight` Borrow the method of hiding the height from TUGboat as well.

```

56 \def\tubhideheight#1{\setbox0=\hbox{#1}%
57   \ht0=0pt \dp0=0pt \box0 }

```

`\XeTeX` Start by defining the first part of X_YT_EX and X_YL^AT_EX.

```

58 \DeclareRobustCommand\Xe[1]{\leavevmode
59   \tubhideheight{\hbox{X%
60     \setbox0=\hbox{\TeX}\setbox1=\hbox{E}%
61     \lower\dp0\hbox{\raise\dp1\hbox{%
62       \kern-.125em\tubreflect{E}}}%
63     \kern-.1667em #1}}%
64 \def\XeTeX{\Xe\TeX}

```

`\XeLaTeX` Define X_YL^AT_EX using the existing macros.

```

65 \def\XeLaTeX{\Xe{\, \LaTeX}}

```

`\LyX` Define the LyX logo.

```

66 \def\LyX{L\kern-.1667em\lower.25em\hbox{Y}\kern-.125emX}

```

`\SMC` Define a new small caps for use in BIB_T_EX, and an error message to go with it (from the ltugboat package).

```

67 \DeclareRobustCommand\SMC{%
68   \ifx\@currsize\normalsize\small\else
69   \ifx\@currsize\small\footnotesize\else
70   \ifx\@currsize\footnotesize\scriptsize\else
71   \ifx\@currsize\large\normalsize\else
72   \ifx\@currsize\Large\large\else
73   \ifx\@currsize\LARGE\Large\else
74   \ifx\@currsize\scriptsize\tiny\else
75   \ifx\@currsize\tiny\tiny\else
76   \ifx\@currsize\huge\LARGE\else
77   \ifx\@currsize\Huge\huge\else
78     \small\SMC@unknown@warning
79   \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
80 }
81 \newcommand\SMC@unknown@warning{\PackageError{classpack}%
82   {\string\SMC: can't find small caps for nonstandard
83     text font size command -- using \string\small}%
84   {Check the font size or scaling for \the\@currsize}}
85 \newcommand\textSMC[1]{\SMC #1}

```

`\BIBTeX` Finally, define `BIBTeX` in various forms.

```

86     \def\Bib{%
87     \ifdim \fontdimen1\font>0pt
88     B{\SMC\SMC IB}%
89     \else
90     \textsc{Bib}%
91     \fi
92   }
93   \def\BibTeX{\Bib\kern-.08em \TeX}
94   \let\BiBTeX\BibTeX
95   \let\BIBTeX\BibTeX

```

The flexlogo package

The flexlogo package (under development) will make this section obsolete, as it allows for the complete redefining of the \TeX , \LaTeX , and related logos for non-CM fonts.

9.9 Formatting additions

`\CPKvstrut` Define a strut that adjusts to the size of type, for use in spacing table headers and footers.

```

96   \newcommand{\CPKvstrut}{\vrule height1.2em
97                               depth.6667ex width0pt}

```

`\CPKmenusep` Define a macro to format an arrow between documentary menu items. Probably no longer needed now that the menukeys package is available

```

98   \def\CPKmenusep{\thinspace$\rightarrow$\thinspace
99                   \allowbreak}

```

`\CPKprestrut` Also a strut to precede paragraph cells...

```

100  \newcommand{\CPKprestrut}{\vrule height1em width0pt}

```

`\CPKpoststrut` ...and one to follow them.

```

101  \newcommand{\CPKpoststrut}{\vrule depth.5ex width0pt}

```

`\hrule` Give the `\hrule` command an optional thickness (height) argument.

```

102  \def\hrule{\noalign{\ifnum0=`}\fi
103    \@ifnextchar[{\@hline}{\@hline[\arrayrulewidth]}}
104  \def\@hline[#1]{\hrule \@height#1 \futurelet
105    \reserved@a\@xhline}

```

`secnumdepth`

```

106  \setcounter{secnumdepth}{5}

```

tocdepth

```
107 \setcounter{tocdepth}{5}
```

IndexColumns

```
108 \@ifundefined{c@IndexColumns}{}{\setcounter{IndexColumns}{2}}
```

columnsep

```
109 \setlength{\columnsep}{3pc}
```

CPKtabwid Measurement of table width, to allow for notes (see figtab2latex.xml).

```
110 \newlength{\CPKtabwid}
```

CPKfnote Define a counter for left-hang notes.

```
111 \newcounter{CPKfnote}
```

\fnote Define a left-hang note to protrude into the left space beside a table cell with an italic number.

```
112 \newcommand{\fnote}{%
113   \stepcounter{CPKfnote}\llap{%
114     \footnotesize\textsuperscript{%
115       \itshape\alph{CPKfnote}\upshape}}\thinspace}}
```

\hyphenation Add some hyphenation oddities.

```
116 \hyphenation{ele-ment ele-ments attri-bute attri-butes
117   docu-ment docu-ments docu-men-tation primi-tive
118   primi-tives name-space name-spaces helico-pter}
```

\descriptionlabel This has been replaced by the enumitem package as specified in prepost.xml.

CPKcoref Add the counter to enable the use of <coref> list counters in generated code.

```
119 \newcounter{CPKcoref}
```

An \endinput command is automatically appended to this file.

10 DocBook DTD shim

This is a customisation layer (shim) for the *DocBook5* DTD. There are no changes to the element structure, only the addition or modification of attributes. The availability of `<html:form>` is closed off by declaring it EMPTY.

This file is in effect a driver, allowing the predeclaration of values *before* the main DTD is read; under the rules of ISO 8859 inherited by XML, this prevents those declarations from being subsequently overwritten when the main file is loaded.

This will at some stage be replaced by a formal modification layer to the RNG version of DocBook.

`cpdtdoc-comment` Identify the file origin, date, time, etc.

```
1 <!-- DocBook DTD shim (from the classpack package v.1.28, 2024-02-21) -->
2 <!-- Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16 -->
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

10.1 Set up the invocation and disable HTML forms

`&db5dtd;` Create a Parameter Entity (PE) by which we can invoke the DTD at the end.

```
3 <!ENTITY % db5dtd SYSTEM
4      "/dtds/docbook/docbook-5.0/dtd/docbook.dtd">
```

`html` Nullify the HTML `<form>`.

```
5 <!ELEMENT html:form EMPTY>
```

`info` Provide an attribute for the cover artwork image.

```
6 <!ATTLIST info coverart CDATA #IMPLIED>
```

10.2 Changes to attribute lists

`@YYYY-MM-DD` Add a human-validatable attribute for the ISO date to the three element types used for dates.

```
7 <!ATTLIST date YYYY-MM-DD CDATA #IMPLIED>
8 <!ATTLIST confdates YYYY-MM-DD-from CDATA #IMPLIED
9      YYYY-MM-DD-to CDATA #IMPLIED>
10 <!ATTLIST year YYYY-MM-DD CDATA #REQUIRED>
```

`@units` Quotations and block quotations already have a `<linkend>` attribute for citing a source, so add units and limits for ranges.

```
11 <!ATTLIST blockquote units CDATA #IMPLIED>
```

```

12             begin CDATA #IMPLIED
13             end CDATA #IMPLIED>
14 <!--ATTLIST quote units CDATA #IMPLIED
15             begin CDATA #IMPLIED
16             end CDATA #IMPLIED>

```

@linkend Allow cross-references to have multiple targets.

```

17 <!--ATTLIST xref linkend IDREFS #REQUIRED>
18 <!--ATTLIST biblioref linkend IDREFS #REQUIRED>

```

@label Add the attributes and data types needed for a functional bibliography entry. The **@label** attribute holds the name of the reference style (a supported biblatex style name). The **@xreflabel** attribute is for the name of any ancillary package required by the style.

```

19 <!--ATTLIST bibliography label CDATA #REQUIRED
20             xreflabel CDATA #IMPLIED>
21 <!--ATTLIST biblioentry xml:id ID #REQUIRED xreflabel
22 (article|book|booklet|collection|inbook|incollection|
23 inproceedings|inreference|manual|mastersthesis|misc|
24 online|patent|periodical|phdthesis|proceedings|
25 reference|report|standard|thesis|bookinbook|suppbook|
26 suppcollection|suppperiodical|mvbook|mvcollection|
27 mvproceedings|mvreference|unpublished)
28 #REQUIRED>
29 <!--ATTLIST biblioid YYYY-MM-DD CDATA #IMPLIED>

```

@role Provide the new data types for annotated code, along with the attributes for the name of the object and the reuse flag.

```

30 <!--ATTLIST annotation role (attribute|attvalue|box|class|
31 color|comment|counter|dtd|element|entity|environment|error|field|
32 file|font|function|language|length|macro|model|option|
33 package|parameter|rubric|setting|switch|template|typeface|
34 variable|UNUSED) #REQUIRED
35             xreflabel CDATA #REQUIRED
36             reuse IDREFS #IMPLIED>
37 <!--ATTLIST programlisting endinglinenumber CDATA #IMPLIED>

```

@version Remark is used for notes about edits and bug fixes.

```

38 <!--ATTLIST remark version CDATA #REQUIRED
39             revision CDATA #REQUIRED>

```

@language Extend the languages used in a dozen or so element types, add an ending line-number for program listings, and add values for the **@role** attribute on the **<varname>** element.

```

40 <!--ENTITY % langs "language (
41   CGI|C|bash|TeX|LaTeX|LaTeX|XML|XSLT|XLS|HTML|XPath|BiBTeX|
42   awk|Java|man|Makefile|METAFONT|DTD) 'LaTeX'">

```

```

43 <!--ATTLIST classname %langs;>
44 <!--ATTLIST cmdsynopsis %langs;>
45 <!--ATTLIST command %langs;>
46 <!--ATTLIST envvar %langs;>
47 <!--ATTLIST function %langs;>
48 <!--ATTLIST literal %langs;>
49 <!--ATTLIST literallayout %langs;>
50 <!--ATTLIST option %langs;>
51 <!--ATTLIST programlisting %langs;
52     endinglinenumber CDATA #IMPLIED>
53 <!--ATTLIST screen %langs;>
54 <!--ATTLIST varname %langs;
55     role (counter|length|prog|color|template|
56         variable|parameter|field|entrytype) #IMPLIED>
57 <!--ATTLIST parameter %langs;>

```

@fontface Add the attributes needed for typographic annotation.

```

58 <!--ATTLIST type fontencoding (T1|OT1|U) #IMPLIED
59     fontface CDATA #IMPLIED
60     fontfamily (roman|sans|monospace) #IMPLIED
61     fontseries (k|l|m|n|b|b|xl|l|s|l|l|c) #IMPLIED
62     fontshape (up|l|t|sc) #IMPLIED
63     fontsize CDATA #IMPLIED
64     fontalign (yes|no) #IMPLIED
65     fontcolor CDATA #IMPLIED>

```

@omit Add the omission attribute to the package specifications and provide type values for the @class attribute on the <tag> element.

```

66 <!--ATTLIST seglistitem omit IDREFS #IMPLIED>
67 <!--ATTLIST tag class (attribute|attvalue|element|emptytag|
68     endtag|declaration|genentity|localname|namespace|
69     numcharref|parameter|pil|prefix|comment|starttag|xmlpi)
70     #IMPLIED>

```

10.3 Parameter Entity switch for character entities

&OUTPUT; Use the conventional PE-based switching mechanism for L^AT_EX vs HTML output (only L^AT_EX implemented so far).

```

71 <!--ENTITY % OUTPUT "LaTeX">
72 <!--ENTITY % %OUTPUT; "INCLUDE">
73 <!--ENTITY % LaTeX "IGNORE">

```

&LaTeX; List the character entities needed for L^AT_EX.

```

74 <!--[%LaTeX;[
75 <!--ENTITY ampers "&#38;#38;">
76 <!--ENTITY BibTeX "&BibTeX{}">
77 <!--ENTITY BibTeX "&BibTeX{}">
78 <!--ENTITY BIBTeX "&BibTeX{}">

```

```

79 <!ENTITY ConTeXt "Con\TeX t{}">
80 <!ENTITY LaTeX "\LaTeX{}">
81 <!ENTITY LaTeX2e "\LaTeXe{}">
82 <!ENTITY XeTeX "\XeTeX{}">
83 <!ENTITY XeLaTeX "\XeLaTeX{}">
84 <!ENTITY LuaTeX "Lua\TeX{}">
85 <!ENTITY LuaLaTeX "Lua\LaTeX{}">
86 <!ENTITY LyX "\LyX{}">
87 <!ENTITY METAFONT "\MF{}">
88 <!ENTITY METAPOST "\MP{}">
89 <!ENTITY TeX "\TeX{}">
90 <!ENTITY bsol "{\texttt{\textbackslash}}">
91 <!ENTITY bsq "\rule{1.2ex}{1.2ex}">
92 <!ENTITY date "\filedate{}">
93 <!ENTITY degree "\textdegree{}">
94 <!ENTITY doctype "\classorpackage{}">
95 <!ENTITY dollar "\$">
96 <!ENTITY hash "\#">
97 <!ENTITY filler "\hfil{}">
98 <!ENTITY frac12 "\nicefrac12">
99 <!ENTITY frac13 "\nicefrac13">
100 <!ENTITY frac23 "\nicefrac23">
101 <!ENTITY hellip "\dots{}">
102 <!ENTITY mdash "~--- ">
103 <!ENTITY mldr "\dotfill{}">
104 <!ENTITY nbsp "~">
105 <!ENTITY ndash "--">
106 <!ENTITY percent "&#x0025;">
107 <!ENTITY square "\raisebox{-1pt}{\Square}">
108 <!ENTITY star "\ensuremath{\ast}">
109 <!ENTITY thinsp "\thinspace{}">
110 <!ENTITY times "x">
111 <!ENTITY specialUuml
112   '{\normalfont"\fontfamily{cdr}\selectfont U}}'>
113 <!ENTITY verbar "\menusep{}">
114 <!ENTITY version "\fileversion{}">
115 <!ENTITY decorule "\decorule{}">
116 <!ENTITY % HTML "IGNORE">
117 ]]>

```

The commands which are non-standard are implemented in the classpack package, or (in the case of `\decorule`) in a package (decorule) implemented by *ClassPack*, and only used in the circumstance of their inclusion.



&HTML; List the character entities needed for HTML. The illegal inclusion of element markup in attribute values renders this invalid, although it is parsed and accepted by Saxon.

```

118 <![%HTML;[
119 <!ENTITY ampers "&#38; amp;">
120 <!ENTITY BiBTeX ' <span class="BiBTeX">B<span

```

```

121         class="I">I</span>B<span class="TEX">T<span
122         class="E">E</span>X</span></span>'>
123 <!ENTITY BibTeX '
124         class="I">I</span>B
125         class="E">E</span>X</span></span>'>
126 <!ENTITY BIBTeX '
127         class="I">I</span>B
128         class="E">E</span>X</span></span>'>
129 <!ENTITY ConTExt 'Con
130         class="E">E</span>X</span>t'>
131 <!ENTITY LaTeX '
132         class="A">A</span><span class="TEX">T
133         class="E">E</span>X</span></span>'>
134 <!ENTITY LaTeX2e '
135         class="A">A</span><span class="TEX">T
136         class="E">E</span>X</span>2&#x3b5;</span>'>
137 <!ENTITY XeTeX '
138         class="E">E</span><span class="TEX">T
139         class="E">E</span>X</span></span>'>
140 <!ENTITY XeLaTeX '
141         class="E">E</span><span class="LATEX">L
142         class="A">A</span><span class="TEX">T
143         class="E">E</span>X</span></span></span>'>
144 <!ENTITY LuaTeX '
145         class="TEX">T
146         class="E">E</span>X</span></span>'>
147 <!ENTITY LuaLaTeX '
148         class="LATEX">L
149         class="A">A</span><span class="TEX">T
150         class="E">E</span>X</span></span></span>'>
151 <!ENTITY LyX '
152         class="Y">Y</span>X</span>'>
153 <!ENTITY METAFONT '
156         class="E">E</span>X</span>'>
157 <!ENTITY bsol "<math>\backslash</math>">
158 <!ENTITY bsq "&#x25A0;">
159 <!ENTITY date "">
160 <!ENTITY degree "°">
161 <!ENTITY doctype "">
162 <!ENTITY dollar "$">
163 <!ENTITY filler "">
164 <!ENTITY frac12 "<math>\frac{1}{2}</math>">
165 <!ENTITY frac13 "&#x2153;">
166 <!ENTITY frac23 "&#x2154;">
167 <!ENTITY hash "&#x23;">
168 <!ENTITY hellip "...">
169 <!ENTITY mdash " − ">
170 <!ENTITY mldr "...">
171 <!ENTITY nbsp " ">
172 <!ENTITY ndash " − ">
173 <!ENTITY percent "&#x0025;">
174 <!ENTITY square "&#x25a1;">

```

```

175 <!ENTITY star "★">
176 <!ENTITY thinsp " ">
177 <!ENTITY times "×">
178 <!ENTITY specialUuml 'Ü'>
179 <!ENTITY verbar "|">
180 <!ENTITY version "">
181 <!ENTITY decorule
182   "<inlinemediaobject><imageobject><imagedata
183     fileref='decorule.png'
184     width='2in'/></imageobject></inlinemediaobject>">
185 ]]>

```

10.4 Invocation

&invocation; Finally, invoke *DocBook*.

```

186   %db5dtd;

```

11 The *db2dtx.xml* script

While the core of your class or package is the *DocBook* XML document containing your \LaTeX code and documentation, the core of the *ClassPack* system is the program or script that turns your XML file into *.dtx* and *.ins* files for distribution as combined code and documentation. This is that script.

The *db2dtx.xml* script is written in XSLT3, a declarative language for processing XML ([WorldWideWeb Consortium XSLT Working Group, 2017](#)). It consists of a set of templates, each of which matches a markup pattern in your XML document. These templates are usually written to match an element type, or an element type in a particular position or with a particular attribute value or subelement, but some are just referenced by name, and act as subroutines which can be reused in many circumstances.

The advantage of using a declarative language is that the script doesn't need to know when and where each element will occur in your file: the XML parser will feed them to the XSLT script as they occur in document order (or as otherwise specified), and the script will then apply any matching template.

The description in this section explains how the script produces your class or package (and other) files.

db2dtxscript-comment Identify the file origin, date, time, etc.

```
1 <!-- The db2dtx.xml script (from the classpack package v.1.28, 2024-02-21) -->
2 <!-- Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16 -->
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

11.1 XML Declaration, namespaces, and setup

<xsl:stylesheet> The script starts with a comment describing the other files on which the code depends. There is then the *<xsl:stylesheet>* start-tag with the namespace declarations:

```
3 <!-- DEPENDENCIES: doctexbook.dtd, docbook5.dtd,
4     db2bibtex.xml, db2md.xml, figtab2latex.xml,
5     rfc2119.xml, classpack.xml, prepost.xml,
6     languages.xml, readme.xml, representation.xml,
7     lppl.xml (or other specified licence).
8     This program accompanies v1.26 of ClassPack.
9 -->
10 <xsl:stylesheet
11   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
12   xmlns:db="http://docbook.org/ns/docbook"
13   xmlns:sil="http://xml.silmaril.ie/identity/functions"
14   xmlns:xlink="http://www.w3.org/1999/xlink"
15   xmlns:xs="http://www.w3.org/2001/XMLSchema"
16   version="3.1"
17   exclude-result-prefixes="xsl db xlink sil xs">
```

The namespaces identify the origin of various features of the script, but for simplicity they are suppressed in the output. Note that this is an XSLT3 script and requires an XSLT3 processor.

`<xsl:output>` This identifies the output methods for *a*) the default output (L^AT_EX is just regarded as text until someone writes a dedicated `latex` method for XSLT); and *b*) several formats for the different ancillary files generated during processing (the use of `<xsl:result-document>` requires a named method separate from the default).

```

18 <xsl:output method="text"/>
19 <xsl:output method="text" name="textFormat"/>
20 <xsl:output method="xml" name="xmlFormat" indent="yes"
21   doctype-system="../doctexbook.dtd"/>
22 <xsl:output method="xml" name="htmlFormat" indent="yes"
23   doctype-system="about:legacy-compat" omit-xml-declaration="yes"/>
24
25 <xsl:strip-space elements="*/>
26
27 <xsl:include href="db2bibtex.xsl"/>
28 <xsl:include href="db2md.xsl"/>
29 <xsl:include href="figtab2latex.xsl"/>

```

We also request the stripping of unwanted white-space from around all elements, and we include three additional XSLT modules, the files *a*) `db2bibtex.xsl` for handling bibliographic formatting; *b*) `db2md.xsl` for outputting the *Markdown* which CTAN requires for the `README.md` file; and *c*) `figtab2latex.xsl` which handles the calculations and positioning required for figures and tables. These are annotated later in this document.

11.2 Parameters and variables

`armour` The `$armour` parameter is the percent-space combination used at the start of lines in the `dtx` file which are destined for the documentation. It could have been hard-coded as a variable, but to provide for other formats in future, it was thought better to pass it as a parameter with the percent-space as the default value.

`shield` The `$shield` variable is used similarly when constructing figures and tables in the `figtab2latex.xsl` module, which can be used independently of *ClassPack*, and is therefore not parameterised.

```

30 <xsl:param name="armour">
31   <xsl:text>% </xsl:text>
32 </xsl:param>
33 <xsl:variable name="shield">
34   <xsl:text>%</xsl:text>
35 </xsl:variable>
36
37 <!-- path to the classpack directory -->
38 <xsl:param name="cpdir"/>
39 <!-- path to the project's app dev directory -->
40 <xsl:param name="appdir"/>

```

`cpdir` The `$cpdir` and `$appdir` parameters are values passed from the command-line or Makefile for *a*) the path to the *ClassPack* installation directory; and *b*) the current class or package's project directory. In a development setting, typically `$cpdir` could be something like `/usr/local/src/classpack` and `$appdir` might be more like `~/texmf/dev/foobar`.

11.2.1 Versioning

`thisversion` The version and version date of *Classpack* are extracted from the currently-installed *ClassPack* documentation XML (used to generate what you are reading now).

```

41 <xsl:variable name="thisversion"
42   select="document(concat($cpdir,
43                         '/classpack/classpack.xml'))
44           /db:book/db:info/db:revhistory/db:revision
45           [not(../db:revision/@version > @version)]
46           /@version"/>
47 <xsl:variable name="thisverdate"
48   select="document(concat($cpdir,
49                         '/classpack/classpack.xml'))
50           /db:book/db:info/db:revhistory/db:revision
51           [not(../db:revision/@version > @version)]
52           /db:date/@YYYY-MM-DD"/>

```

`thisversion` **TODO: Move version and date elsewhere**

11.2.2 Lookup files

There are five variables defined as reference lookups into other files which **MUST** be present for this script to work:

- `prepost` 1. `$prepost` is the `prepost.xml` file of package data
- `langs` 2. `$langs` is a file `languages.xml` of languages supported by babel
- `readme` 3. `$readme` is the `readme.xml` template for generating `README.md`
- `licence` 4. `$licence` is the software licence or copyright text XML file (eg `lpp1.xml`) which gets included at the end of the documentation
- `thisdoc` 5. `$thisdoc` is a global name for the whole *ClassPack* document being processed so that it can be referred to from within the scope of a different node-set such as an *XPath* tokenize loop or the processing of an external document.

```

53 <xsl:variable name="prepost"
54   select="document(concat($cpdir, '/prepost.xml'))
55   /db:refsection"/>
56 <xsl:variable name="langs"
57   select="document(concat($cpdir, '/languages-master.xml'))
58   /languages"/>
59 <xsl:variable name="readme"
60   select="document(concat($cpdir, '/readme-master.xml'))
61   /db:chapter"/>
62 <xsl:variable name="licence"

```

```

63     select="document(concat($appdir, '/',
64                           /db:book/@audience, '.xml'))
65                           /db:chapter"/>
66     <xsl:variable name="thisdoc" select=""/>

```

11.2.3 Naming

`maxcodelen` We define variables `$maxcodelen`, used in the breakline named template (see the
`breakline` code on page 274) which handles line-length for plaintext output; and `$docname`,
`docname` which is the `@xml:id` of the *ClassPack* document being processed.

```

67     <xsl:variable name="maxcodelen">
68         <xsl:text>40</xsl:text>
69     </xsl:variable>
70
71     <xsl:variable name="docname" select="/db:book/@xml:id"/>
72     <xsl:variable name="doctitle">
73         <xsl:call-template name="detex">
74             <xsl:with-param name="string"
75                 select="/db:book/db:info/db:title/node()"/>
76         </xsl:call-template>
77     </xsl:variable>
78
79     <xsl:variable name="firstverdate"
80         select="/db:book/db:info/db:revhistory/db:revision
81             [not(../db:revision/@version < @version)]
82             /db:date/@YYYY-MM-DD"/>

```

11.2.4 Document metadata

`name` The `$name` of the output document may be affected by the value of the `@xml:id` of the first chapter element in the ‘code’ part; otherwise it’s the document `<ID>` extracted above.

```

83     <xsl:variable name="name">
84         <xsl:choose>
85             <!-- if there is more than one code appendix,
86                  use the ID of the first chapter in the code part
87                  as the 'name' of this bundle -->
88             <xsl:when
89                 test="count(/db:book/db:part[@xml:id='code']
90                     //db:appendix
91                     [substring-after(@xlink:href, '.')
92                     = 'cls'])>1">
93                 <xsl:value-of
94                     select="/db:book/db:part[@xml:id='code']
95                         /db:chapter[1]/@xml:id"/>
96             </xsl:when>
97             <!-- otherwise use the ID of this document -->
98             <xsl:otherwise>
99                 <xsl:value-of select="$docname"/>
100             </xsl:otherwise>
101         </xsl:choose>
102     </xsl:variable>

```

Extract the document metadata from the attributes of the <book> (root) element. These are explained in detail in [section 3.3 on page 21](#).

```

103 <xsl:variable name="doctype" select="/db:book/@arch"/>
104 <xsl:variable name="version" select="/db:book/@version"/>
105 <xsl:variable name="revision" select="/db:book/@revision"/>
106 <xsl:variable name="filetype" select="/db:book/@userlevel"/>
107 <xsl:variable name="date"
108     select="/db:book/db:info/db:revhistory/db:revision
109           [not(../db:revision/@version > @version)]
110           /db:date/@YYYY-MM-DD"/>
111 <xsl:variable name="latestrevhist"
112     select="/db:book/db:info/db:revhistory/db:revision
113           [not(../db:revision/@version > @version)]
114           /@version"/>

```

11.2.5 Text control

`commentchars` A comment character is declared, selected from the `$commentchars` variable: in most cases (L^AT_EX code) this is the % sign, but *ClassPack* can also be used to maintain and manage Bash shell scripts using #.

`commentchars` We set up a variable of assorted comment characters in `$commentchars` that we can use for dividing sections of a generated file.

`sentinel` On that basis, also set up a `$sentinel` variable consisting of two comment characters and a space, for use in adding inline (text) documentation to an extracted document.

```

115 <!-- comment characters depend on file type
116       so extractable chapters/appendixes MUST have
117       an xlink:href ending with one of these filetypes -->
118 <xsl:variable name="commentchars">
119     <class    filetype="cls">%</class>
120     <document filetype="bib">%</document>
121     <document filetype="tex">%</document>
122     <document filetype="cnf">%</document>
123     <document filetype="def">%</document>
124     <document filetype="dtd" term="--&gt;">&lt;!--</document>
125     <document filetype="xml" term="--&gt;">&lt;!--</document>
126     <package  filetype="sty">%</package>
127     <script   filetype="cp">#</script>
128     <script   filetype="sh">#</script>
129     <script   filetype="awk">#</script>
130     <script   filetype="xsl" term="--&gt;">&lt;!--</script>
131 </xsl:variable>
132
133 <!-- Default shield is double delimiter plus space -->
134 <xsl:variable name="sentinel"
135     select="concat(
136         $commentchars/*[name()='current()/db:book/@arch]
137         [@filetype=current()/db:book/@userlevel],
138         $commentchars/*[name()='current()/db:book/@arch]
139         [@filetype=current()/db:book/@userlevel], ' '"/>

```

11.2.6 Process control

The default \LaTeX processor for the documentation or the class/package is still *pdflatex* but the variable `$latexprog` could be set to the value specified in the `@arch` attribute of the relevant package list (see [section 4.1 on page 39](#)). This has been superseded by placing the processor name in the `@conformance` attribute of the relevant part element (see the note in [section 3.3 on page 23](#)).

```

140   <xsl:variable name="latexprog">
141     <xsl:choose>
142       <xsl:when
143         test="/db:book/db:info/db:cover/db:constraintdef
144           [@xml:id=concat(/db:book/@userlevel,'packages')]
145           /@arch!=''">
146         <xsl:value-of
147           select="/db:book/db:info/db:cover
148             /db:constraintdef
149             [@xml:id=concat(/db:book/@userlevel,'packages')]
150             /@arch"/>
151       </xsl:when>
152       <xsl:otherwise>
153         <xsl:text>pdflatex</xsl:text>
154       </xsl:otherwise>
155     </xsl:choose>
156   </xsl:variable>

```

TODO: Remove unused traces of the build system The specification of the processor is now kept elsewhere. Delete them

11.3 Creating the .dtx file structure

This is the template matching the document root, where all the preparatory work is done.

Start with the program identity in the normal way, then check that version and revision in the root element match the latest revision history

```

157   <xsl:template match="/">
158     <xsl:message>
159       <xsl:text>This is DB2DTX, Version </xsl:text>
160       <xsl:value-of select="$thisversion"/>
161       <xsl:text> (</xsl:text>
162       <xsl:value-of select="$thisverdate"/>
163       <xsl:text>).</xsl:text>
164     </xsl:message>
165     <xsl:variable name="fence">
166       <xsl:call-template name="makefence"/>
167     </xsl:variable>
168     <!-- stop if the version numbers don't accord -->
169     <xsl:choose>
170       <xsl:when test="concat($version,'.', $revision)
171         != $latestrevhist">

```

```

172     <xsl:message>
173       <xsl:text>! Declared version </xsl:text>
174       <xsl:value-of
175         select="concat($version, '.', $revision)"/>
176       <xsl:text> does not match latest </xsl:text>
177       <xsl:text>revision history </xsl:text>
178       <xsl:value-of select="$latestrevhist"/>
179       <xsl:text>&#xa; I'm sorry, </xsl:text>
180       <xsl:text>I can't go on </xsl:text>
181       <xsl:text>until you fix this.</xsl:text>
182     </xsl:message>
183   </xsl:when>

```

11.3.1 Installer, README, and MANIFEST

Call templates to create the .ins file, the README file, and the MANIFEST file. The first is done via the template for the <info> element (see [section 11.6 on page 281](#)), the other two by named templates (see [section 11.5.3 on page 261](#) and [section 11.5.3 on page 262](#)).

```

184   <xsl:otherwise>
185     <!-- output the .ins file first -->
186     <xsl:apply-templates
187       select="/db:book/db:info" mode="ins"/>
188     <!-- then the README.md -->
189     <xsl:call-template name="readme"/>
190     <!-- and then the MANIFEST -->
191     <xsl:call-template name="manifest"/>
192     <!-- and then the index.html for the Silmaril site -->
193     <xsl:call-template name="webindex"/>

```

TODO: Makefile needs to manage ancillary and test files properly Find a way to test the code and pass conditions to the Makefile for other files

11.3.2 Copyright

The .dtx file starts with the switch to enable commenting, and the boilerplate of the copyright statement, which is called as a named template (see [section 11.5.5 on page 276](#)).

```

194   <!-- start creating the .dtx file -->
195   <xsl:message>
196     <xsl:text>Making </xsl:text>
197     <xsl:value-of select="$doctype"/>
198     <xsl:text> </xsl:text>
199     <xsl:value-of select="$docname"/>
200   </xsl:message>
201   <xsl:text>% \iffalse meta-comment&#xa;%&#xa;</xsl:text>
202   <xsl:call-template
203     name="copyright-statement">
204     <xsl:with-param name="ftype">
205       <xsl:text>dtx</xsl:text>
206     </xsl:with-param>
207   </xsl:call-template>

```

```

208         <xsl:text>%
209 % \fi
210 % \iffalse&#xa;</xsl:text>

```

11.3.3 Special for class files

Document classes need a \ProvidesFile command.

```

211         <!-- SPECIAL EXCEPTION FOR CLASSES:
212             needs \ProvidesFile -->
213         <xsl:if test="$doctype='class'">
214             <xsl:text>%&lt;*driver>
215 \ProvidesFile{</xsl:text>
216             <xsl:value-of select="$docname"/>
217             <xsl:text>.dtx}
218 %&lt;/driver>
219 </xsl:text>
220         </xsl:if>

```

11.3.4 Special for shell scripts

As mentioned in [section 11.2.5 on page 95](#), *ClassPack* can also be used to manage and maintain shell scripts. This section deals with the extra code needed to generate the header of a script.

11.3.4.1 Shell call : Construct the ‘shebang’ line at the top with the path to the shell binary. The first line we output is the DocTeX tag to start extraction, followed by the shebang.

```

221         <xsl:choose>
222         <!-- SPECIAL EXCEPTION FOR SHELL SCRIPTS: -->
223         <xsl:when test="$doctype='script'">
224             <!-- shell call -->
225             <xsl:text>%&lt;</xsl:text>
226             <xsl:value-of select="$doctype"/>
227             <xsl:text>></xsl:text>
228             <xsl:value-of select="$commentchars"/>
229             <xsl:text>! /bin/</xsl:text>
230             <xsl:value-of select="/db:book/@userlevel"/>
231             <xsl:text>&#xa;</xsl:text>

```

11.3.4.2 Script author : Next comes the author and date.

```

232         <!-- author, affil, date -->
233         <xsl:text>%&lt;</xsl:text>
234         <xsl:value-of select="$doctype"/>
235         <xsl:text>></xsl:text>
236         <xsl:value-of select="$fence"/>
237         <xsl:text>%&lt;</xsl:text>
238         <xsl:value-of select="$doctype"/>
239         <xsl:text>></xsl:text>
240         <xsl:value-of select="$sentinel"/>
241         <xsl:value-of

```

```

242         select="/db:book/db:info/db:author[1]
243             /db:personname/db:firstname"/>
244     <xsl:text> </xsl:text>
245     <xsl:value-of
246         select="/db:book/db:info/db:author[1]
247             /db:personname/db:surname"/>
248     <xsl:text>, </xsl:text>
249     <xsl:value-of
250         select="/db:book/db:info/db:author[1]
251             /db:affiliation/db:orgname"/>
252     <xsl:text>, </xsl:text>
253     <xsl:value-of
254         select="format-date(
255             /db:book/db:info
256             /db:revhistory/db:revision
257             [@version=max(parent::db:revhistory
258                 /db:revision/@version)]
259             /db:date/@YYYY-MM-DD,
260             '[D] [Mn] [Y]')"/>
261     <xsl:text>&#xa;</xsl:text>

```

11.3.4.3 Script history: Output the development and maintenance history

```

262     <!-- history -->
263     <xsl:text>%&lt;</xsl:text>
264     <xsl:value-of select="$doctype"/>
265     <xsl:text>></xsl:text>
266     <xsl:value-of select="$fence"/>
267     <xsl:for-each
268         select="/db:book/db:info
269             /db:revhistory/db:revision">
270         <xsl:sort select="@version"
271             order="ascending"/>
272         <xsl:text>%&lt;</xsl:text>
273         <xsl:value-of select="$doctype"/>
274         <xsl:text>></xsl:text>
275         <xsl:value-of select="$sentinel"/>
276         <xsl:value-of select="@version"/>
277         <xsl:text> </xsl:text>
278         <xsl:value-of
279             select="db:revdescription/db:itemizedlist
280                 /db:title"/>
281         <xsl:text>: </xsl:text>
282         <xsl:for-each
283             select="db:revdescription/db:itemizedlist
284                 /db:listitem">
285             <xsl:value-of select="normalize-space()"/>
286         </xsl:for-each>
287         <xsl:text> </xsl:text>
288         <xsl:value-of select="db:date/@YYYY-MM-DD"/>
289         <xsl:text>&#xa;</xsl:text>
290     </xsl:for-each>

```

11.3.4.4 Script version : Maintain the version data and output.

```

291      <xsl:text>%&lt;</xsl:text>
292      <xsl:value-of select="$doctype"/>
293      <xsl:text>></xsl:text>
294      <xsl:value-of select="$fence"/>
295      <!-- version and date -->
296      <xsl:text>%&lt;</xsl:text>
297      <xsl:value-of select="$doctype"/>
298      <xsl:text>>VERSION=</xsl:text>
299      <xsl:value-of
300        select="/db:book/db:info
301              /db:revhistory/db:revision
302              [@version=max(parent::db:revhistory
303                          /db:revision/@version)]
304              /@version"/>
305      <xsl:text>&#xa;</xsl:text>
306      <xsl:text>%&lt;</xsl:text>
307      <xsl:value-of select="$doctype"/>
308      <xsl:text>>VERSDATE=</xsl:text>
309      <xsl:value-of
310        select="/db:book/db:info
311              /db:revhistory/db:revision
312              [@version=max(parent::db:revhistory
313                          /db:revision/@version)]
314              /db:date/@YYYY-MM-DD"/>
315      <xsl:text>&#xa;</xsl:text>

```

11.3.4.5 Script name : Identify the script.

```

316      <xsl:text>%&lt;</xsl:text>
317      <xsl:value-of select="$doctype"/>
318      <xsl:text>>PROGRAM=`echo $0 |</xsl:text>
319      <xsl:text>awk -F/ '{print $NF}' |</xsl:text>
320      <xsl:text>awk -F. '{print $1}'`</xsl:text>
321      <xsl:text>&#xa;</xsl:text>
322      <xsl:text>%&lt;</xsl:text>
323      <xsl:value-of select="$doctype"/>
324      <xsl:text>></xsl:text>
325      <xsl:value-of select="$fence"/>
326    </xsl:when>
327    <!-- end of block for shell scripts -->

```

11.3.5 File header

Other document types (classes or packages) need the DocT_EX ‘tags’ which delimit code extraction, and the T_EX format control limits. This is raw code generated automatically, not something derived from the documentation. It also needs to be called by any appendix that contains an extractable code file (eg a style accompanying a class).

11.3.5.1 Document type tag : We create the named start-tag for code extraction, plus the minimum (earliest-dated) L^AT_EX format needed to process it, which

we get from the the document start-tag attributes @conformance and @condition.

```

328      <!-- ELSE FOR PACKAGES AND CLASSES -->
329      <xsl:otherwise>
330        <xsl:text>%&lt;</xsl:text>
331        <xsl:value-of select="$doctype"/>
332        <xsl:text>>\NeedsTeXFormat{</xsl:text>
333        <xsl:value-of select="/db:book/@conformance"/>
334        <xsl:text>}</xsl:text>
335        <xsl:value-of
336          select="translate(/db:book/@condition,
337                        '-','/')"/>
338        <xsl:text>]&#xa;</xsl:text>

```

Generate the end-tag; then emit the \ProvidesClass or \ProvidesPackage command using the name of the document, but allowing an override with the ID of the first code chapter.

This override has been temporarily suppressed as it causes a package naming conflict in certain circumstances,

Add the date and the title of the document (with L^AT_EX logos replaced by plain text), repeating the tag for a second line.

```

339      <xsl:text>%&lt;</xsl:text>
340      <xsl:value-of select="$doctype"/>
341      <xsl:text>>\Provides</xsl:text>
342      <xsl:value-of
343        select="upper-case(substring($doctype,1,1))"/>
344      <xsl:value-of select="substring($doctype,2)"/>
345      <xsl:text>{</xsl:text>
346      <xsl:value-of select="$docname"/>
347      <xsl:text>}</xsl:text>
348      <!-- use latest rev date as distro date -->
349      <xsl:value-of
350        select="translate($date,'-','/')"/>
351      <xsl:text> v</xsl:text>
352      <xsl:value-of select="/db:book/@version"/>
353      <xsl:text>.</xsl:text>
354      <xsl:value-of select="/db:book/@revision"/>
355      <xsl:text>&#xa;%&lt;</xsl:text>
356      <xsl:value-of select="$doctype"/>
357      <xsl:text>> </xsl:text>
358      <xsl:variable name="title">
359        <xsl:choose>
360          <!-- use TeX-parsed content for code title -->
361          <xsl:when test="//db:part[@xml:id='code']
362                        /db:title">
363            <xsl:value-of
364              select="normalize-space(
365                replace(
366                  //db:part[@xml:id='code']
367                  /db:title,
368                  '\LaTeX\{\}', 'LaTeX'))"/>

```

```

368         </xsl:when>
369         <xsl:when test="//db:info/db:subtitle">
370             <xsl:value-of
371                 select="normalize-space(
372                     //db:info/db:subtitle)"/>
373         </xsl:when>
374         <xsl:otherwise>
375             <xsl:value-of
376                 select="normalize-space(
377                     //db:info/db:title)"/>
378         </xsl:otherwise>
379     </xsl:choose>
380 </xsl:variable>
381 <xsl:call-template name="delogify">
382     <xsl:with-param name="string"
383         select="$title"/>
384 </xsl:call-template>
385 <xsl:text>]&#xa;</xsl:text>
386 </xsl:otherwise>
387 </xsl:choose>

```

11.3.5.2 Constructing the list of packages : The autoloading of a package is specified in `prepost.xml` by one of six methods:

1. *by default*, using a null value for `<constructorsynopsis>/@condition`.

```

388     <step condition="doc" remap="makeidx">
389         <para>Package for creating indexes.</para>
390         <constructorsynopsis condition=""/>
391     </step>

```

In this example, the `makeidx` package will be included in the documentation for every *ClassPack* class or package produced.

2. *by match*, using a `<constructorsynopsis>/@condition` whose value is the local-name of a *DocBook* element type which occurs in the documentation section of the *ClassPack* document being processed.

```

392     <step remap="url" condition="doc">
393         <para>Handling of URI formatting.</para>
394         <constructorsynopsis condition="link"/>
395         <constructorsynopsis condition="uri"/>
396         <constructorsynopsis condition="email"/>
397     </step>

```

In this example, the `url` package is included if a `link`, `uri`, or `email` element occurs in the documentation.

An optional `<methodparam>` element with `<parameter>` and `<modifier>` subelements can be used to specify what attribute and value on the identified element to test for inclusion. For example,

```

<step remap="array" condition="cls sty doc">
    <constructorsynopsis condition="tgroup"/>

```

```
</step>
```

means ‘add the array package if the documentation uses the `<tgroup>` element type’ — because if an author uses tables, it’s likely that the features of the array package will be needed. A more complex example is:

```
<step condition="cls sty doc" remap="csquotes">
  <para>Adds correct curly quotes for cited titles when using
    <package>biblatex</package></para>
  <constructorsynopsis condition="bibliography">
    <methodparam>
      <parameter>arch</parameter>
      <modifier>biblatex</modifier>
    </methodparam>
  </constructorsynopsis>
</step>
```

meaning ‘add the csquotes package if there is a `<bibliography>` element with an `@arch` attribute set to “biblatex”’.

We go through the PRE and POST halves of `prepost.xml` and get every subelement `<constructorsynopsis>` of those `<step>` elements (packages) which are tagged for “doc” usage.

```
398      <!-- AUTOPACKAGE STARTS HERE -->
399      <!-- 1. CREATE THE LIST OF PACKAGES NEEDED FOR DOCUMENTATION -->
400      <xsl:variable name="packages">
401        <!-- Create a list of <seglistitem>s -->
402        <xsl:call-template name="makepackages"/>
403      </xsl:variable>
404      <!-- 2. OUTPUT THE LIST FOR REFERENCE, unsorted and unpruned -->
405      <xsl:result-document format="xmlFormat"
406        href="autopack.xml">
407        <db:segmentedlist arch="{ $doctype}" remap="{ $docname}">
408          <xsl:copy-of select="$packages"/>
409        </db:segmentedlist>
410      </xsl:result-document>
```

11.3.5.3 Preloaded packages and options : At this stage we are ready to work out what packages are needed, both author-specified and automated; *but not* those marked for deferral to a later section identified by an ID in `@linkend`. This is done to allow some package declarations to wait until after options are processed and perhaps a base class loaded.

```
411      <!-- 3. NON-DEFERRED PACKAGES (u/s) -->
412      <xsl:call-template name="nondeferredpackages"/>
413      <!-- unsure about them -->
414      <!-- 4. START THE CODE FOR THE DOCUMENTATION (the "driver") -->
415      <xsl:text>%&lt;*driver>&#xa;</xsl:text>
```

This marks the end of the common preamble (for scripts and \LaTeX classes or packages) — from here on we work on the documentation.

11.3.5.4 Special treatment for packages needing preloading : A few packages might need preloading. The only common one is fix-cm package, which is still needed to liberate font size changes from the original Computer Modern step-size defaults, even with X_{\LaTeX} and $\text{Lua}\LaTeX$. This can be avoided by using the `\RequirePackage` command *at the start of the document*, before the `\documentclass` command.

The fix-cm package is therefore still included by default in the author's own packages. Regardless of the typeface an author chooses for documentation, someone may want to reproduce the documentation using Computer Modern and may still be using the fontenc and inputenc packages in their *ClassPack*-generated code.

The anysizefont package may supersede fix-cm at some future stage, as that package does not require preloading.
The fixltx2e package is no longer required for preloading, as all its provisions are now included in the \LaTeX kernel.

```

416      <!-- 5a. do the compulsory preloads first -->
417      <xsl:for-each select="$packages/db:seglistitem
418                  [db:seg/@annotations='required']">
419          <!-- note that the inverse of this condition is needed
420               later to PREVENT this package being loaded on a
421               second or subsequent time -->
422          <xsl:text>\RequirePackage</xsl:text>
423          <xsl:if test="db:seg/@role!=''">
424              <xsl:text>[</xsl:text>
425              <xsl:value-of select="db:seg/@role"/>
426              <xsl:text>]</xsl:text>
427          </xsl:if>
428          <xsl:text>{</xsl:text>
429          <xsl:value-of select="db:seg"/>
430          <xsl:text>}% </xsl:text>
431          <xsl:text>because "</xsl:text>
432          <xsl:value-of select="@role"/>
433          <xsl:text>" :: "</xsl:text>
434          <xsl:value-of select="@arch"/>
435          <xsl:text>"&#xa;</xsl:text>
436      </xsl:for-each>
437      <!-- end of compulsory preloads -->

```

11.3.5.5 Special treatment for conflicting package options : The error message ‘Option clash for package ...’ means that some other package being loaded needs different options to any the author has specified (or not). This can be avoided by using the `\PassOptionsToPackage` command *at the start of the document*, before the `\documentclass` command.

Possibly the most common occurrence is with `xcolor`, where the author wants one palette loaded and `hyperref` wants another, for example

1. the **svgnames** option to the `xcolor` package can be passed to `xcolor` even if it is loaded later with a different option requested
2. similarly, the **hyphens** option to the `url` package (more of a convenience to authors than a specific requirement) can be passed to the `url` package even if it is loaded later with no options requested

```

438      <!-- @arch here is the class of autopackage mapping -->
439      <!--      |
440              $packages/db:seglistitem
441              [db:seg=$thisdoc//db:constraintdef
442              [xml:id='stypackages']//db:seg[@role]]
443      this was removed to stop it affecting sty packages -->
444      <xsl:for-each-group
445      select="$packages/db:seglistitem
446      [@arch='dependencies']
447      [db:seg/@annotations='optional']
448      |
449      $packages/db:seglistitem
450      [@arch='requests']
451      [db:seg[@role]=$packages/db:seglistitem
452      [@arch!='requests']/db:seg]
453      |
454      $packages/db:seglistitem
455      [@arch='defaults']
456      [db:seg=$prepost/db:procedure
457      [xml:id='prepackage']
458      /db:step[@role and @performance]
459      /@remap]"
460      group-by="db:seg">
461      <xsl:text>\PassOptionsToPackage{</xsl:text>
462      <!-- use the @role on the <seg>, at least one of them has one
463      but create them as a variable so we can group them -->
464      <xsl:variable name="options">
465      <xsl:for-each select="current-group()">
466      <xsl:if test="db:seg/@role">
467      <xsl:for-each select="tokenize(db:seg/@role,' ')">
468      <opt>
469      <xsl:value-of select="."/>
470      </opt>
471      </xsl:for-each>
472      </xsl:if>
473      <xsl:if test="@arch='dependencies'
474      and @audience
475      and @role">
476      <xsl:for-each select="tokenize(@role,' ')">
477      <opt>
478      <xsl:value-of select="."/>
479      </opt>
480      </xsl:for-each>
481      </xsl:if>

```

```

482         </xsl:for-each>
483         <!-- also use the @role in the <seglistitem> -->
484         <xsl:for-each select="tokenize(@role,' ')">
485             <opt>
486                 <xsl:value-of select="."/>
487             </opt>
488         </xsl:for-each>
489     </xsl:variable>
490     <xsl:for-each-group select="$options" group-by="opt">
491         <xsl:if test="position()>1">
492             <xsl:text>,</xsl:text>
493         </xsl:if>
494         <xsl:value-of select="current-grouping-key()"/>
495     </xsl:for-each-group>
496     <!-- say what -->
497     <xsl:message>
498         <xsl:text>Passing options "</xsl:text>
499         <xsl:for-each-group select="$options" group-by="opt">
500             <xsl:if test="position()>1">
501                 <xsl:text>,</xsl:text>
502             </xsl:if>
503             <xsl:value-of select="current-grouping-key()"/>
504         </xsl:for-each-group>
505         <xsl:text>" to package </xsl:text>
506         <xsl:value-of select="db:seg"/>
507     </xsl:message>
508     <xsl:text>}{</xsl:text>
509     <xsl:value-of select="db:seg"/>
510     <xsl:text>}% because </xsl:text>
511     <xsl:value-of select="@arch"/>
512     <xsl:text> </xsl:text>
513     <xsl:value-of select="@audience | @condition"/>
514     <xsl:text> requires </xsl:text>
515     <xsl:value-of select="@role"/>
516     <xsl:text>&#xa;</xsl:text>
517 </xsl:for-each-group>
518 <!-- end of 5b. passing options to packages -->
519 <!-- 6. ADD COMMANDS to hold the docname and doctype -->
520 <xsl:text>\providecommand{\CPKdocname}{</xsl:text>
521 <xsl:value-of select="/db:book/@xml:id"/>
522 <xsl:text>}&#xa;\providecommand{\CPKdoctype}{</xsl:text>
523 <xsl:value-of select="/db:book/@arch"/>
524 <xsl:text>}&#xa;</xsl:text>

```

11.3.5.6 Language preloads for the babel package : The babel package needs individual attention because the languages need to be preloaded, and it interacts with biblatex as well. The base language of the document comes last, so we record it in a variable here, then fish out any other languages, and list them into the argument of `\PassOptionsToPackage`, followed by the base language. The console message identifies the languages by human name from the node-set in `$langs`, taken from the ancillary file `languages.xml`.

```

525 <!-- SPECIAL FOR BABEL, other languages preloaded
526      This kind of dependency can't be expressed in
527      prepost.xml -->
528 <xsl:variable name="baselang">
529   <xsl:choose>
530     <xsl:when test="/db:book/@xml:lang">
531       <xsl:value-of select="/db:book/@xml:lang"/>
532     </xsl:when>
533     <xsl:otherwise>
534       <xsl:text>en-IE</xsl:text>
535     </xsl:otherwise>
536   </xsl:choose>
537 </xsl:variable>
538 <!-- see what other langs are used: only in the 'doc'
539      part and its descendants; if @xml:lang is
540      given *and* if it's non-null *and* if it's
541      not the $baselang we already snagged above -->
542 <xsl:if test="/db:book/db:part[@xml:id='doc']
543      /descendant-or-self::*[@xml:lang]
544      [@xml:lang!='']
545      [@xml:lang!=$baselang]
546      or
547      /db:book/@xml:lang">
548   <xsl:text>\PassOptionsToPackage{</xsl:text>
549   <xsl:variable name="loadedlangs">
550     <xsl:for-each-group group-by="."
551       select="/db:book/db:part[@xml:id='doc']
552       /descendant-or-self::*[@xml:lang]
553       [@xml:lang!='']
554       [@xml:lang!=$baselang]/@xml:lang">
555       <xsl:value-of
556         select="$langs/language
557           [(string-length(current-grouping-key())>2
558             and concat(@Lang,'-',@Country)
559             =current-grouping-key())
560           or
561             (string-length(current-grouping-key())=2
562               and
563               @Lang=current-grouping-key())]
564           [position()=last()]/@BabelName"/>
565       <xsl:text>,</xsl:text>
566     </xsl:for-each-group>
567     <xsl:value-of
568       select="$langs/language
569         [(string-length($baselang)>2
570           and concat(@Lang,'-',@Country)
571           =$baselang)
572         or
573           (string-length($baselang)=2
574             and
575             @Lang=$baselang)]
576         [position()=last()]/@BabelName"/>
577   </xsl:variable>
578   <xsl:value-of select="$loadedlangs"/>

```

```

579         <xsl:text>}{babel}% use of other (non-</xsl:text>
580         <xsl:value-of select="$baselang"/>
581         <xsl:text>) languages&#xa;</xsl:text>
582     </xsl:if>

```

11.3.5.7 The `ltxdoc \documentclass`: Finally, output the `\documentclass` command for `ltxdoc`, with any options taken from the `@remap` attribute on the document root element.

```

583     <xsl:text>\documentclass</xsl:text>
584     <xsl:if test="/db:book/@remap">
585         <xsl:text>[</xsl:text>
586         <xsl:value-of select="/db:book/@remap"/>
587         <xsl:text>]</xsl:text>
588     </xsl:if>
589     <xsl:text>{ltxdoc}&#xa;</xsl:text>

```

11.3.6 Automated package management

packages *AutoPackage* creates a node-set variable called `$packages` listing *all* the packages required *for the documentation*, both those detected by settings in `prepost.xml` and those specified manually in the document by the author.

This node-set is in the form of a `<seglistitem>` for each package — the same format as used in a *ClassPack* file for declaring the author-required packages (see [section 4.1 on page 39](#)) but using additional attributes to control the loading.

```

<seglistitem
  arch="WHY IT'S BEING INSERTED"
  remap="EXCLUSIONS"
  conformance="{@conformance}"
  condition="{ 'off'=DO NOT LOAD}"
  userlevel="LOCATION IN PREPOST.XML"
  annotations="DEPENDENCIES">
  <seg version="YYYY-MM-DD"
    condition="DOC|CLS|STY"
    role="OPTIONS">PACKAGENAME</seg>
</seglistitem>

```

The capitalised mnemonics in this template example represent the following values brought over from *a*) the `prepost.xml` database; and *b*) the author's requests in the document. The attributes on the `seglistitem` are:

1. `@arch` is the *AutoPackage* mapping class in which the entry in `prepost.xml` was found that matched, and resulted in this package being marked for inclusion;
2. `@remap` is a list of package names that will need to be excluded if the current package is loaded; for example, `inputenc` and `fontenc` are excluded if `fontspec` is selected (the default anyway);

3. @annotations is a list of external dependencies that say how the document must be processed, like "xelatex" or "biblatex".
4. @conformance is a copy of the @conformance attribute on a step element in prepost.xml signalling that additional code may be required (see [section 5.3 on page 51](#)). This is experimental and may be dropped or superseded.
5. @condition if set to "off", the package will not be loaded. Not currently used.
6. @userlevel is a decimal number signifying the location in prepost.xml where the packaga data was found. The integer part should always be 1, representing the first procedure element (pre-package); the decimal part is the numeric location of the step element within that procedure. This is used to sort the selected packages before output, because order in prepost.xml is *significant*.

The attributes on the seg element are:

1. @version (optional) the version of the package needed to support the document being written;
2. @role (optional) any options needed for loading this package;
3. @condition is the list of document types ("doc", "cls", or "sty") from prepost.xml ;
4. The text content of the seg element is the name of the package.

packages The whole tree of the \$packages variable (all the generated seglistitems is output to the file autopack.xml as a log of what was used. This is useful when tracing oddities like packages being (or not being) included (or excluded).

11.3.6.1 Implement the packages : If, after all that, there is something to do, output the \RequirePackage commands needed.

packages Go through each package recorded in \$packages. There may be many matches (multiple ways for a package to be included), but we only want to load the package once, so we group them by name, and sort them by their location in prepost.xml ; this lets us control the order (eg the dox package always comes first).

While we're doing this, we test for how one package can be used to prevent another one from loading (conflicted inclusion): we exclude any package mentioned in the <constructorsynopsis>/@remap of *another* package in the list; this lets us exclude, for example, inputenc and fontenc when fontspec has been selected by the author. We also test for packages to be *excluded* on the grounds of their @conformance value.

The inclusion code is generated by the named template <packages> (it gets called from elsewhere in the script as well).

```
590      <!-- RATIONALISATION OF ANY PACKAGES DETECTED -->
591      <xsl:if test="count($packages/db:seglistitem)>0">
```

```

592      <!-- They need to be handled in the order in which they occur
593           in prepost.xml because that determines the sequence in
594           which packages get loaded. Because packages may occur
595           more than once in $packages (eg loaded by deduction,
596           and loaded by request) we can use grouping to make sure
597           multiple entries get handled together. The order from
598           prepost.xml is carried in the @userlevel attribute BUT
599           this used to be overridden for any packages requested
600           by the author, in order to ensure they come last, but
601           this is probably A Bad Idea and has been reverted,
602           see under APauthorreq -->
603      <xsl:text>%%
604      %% Packages for documentation
605      %%&#xa;</xsl:text>
606      <!-- sort by package userlevel for the order from prepost.xml
607           because it's unique, although a package can still get
608           specified multiple times, so within it, sort by
609           @security, which hold the package name requested by
610           the author -->
611      <!-- grouping by @userlevel lets us identify duplicates -->
612      <xsl:for-each-group select="$packages/db:seglistitem"
613           group-by="@userlevel">
614        <xsl:sort select="@security"/>
615        <!-- maxlevel should no longer be needed -->
616        <xsl:variable name="maxlevel"
617           select="current-group()
618               [not(current-group()/@userlevel
619                   &lt; @userlevel)][1]/@userlevel"/>
620      <xsl:choose>
621        <!-- OMIT if another entry (not this one)
622             has this package name specified in @remap
623             eg fontspec will mention inputenc and fontenc
624             both of which need omitting if fontspec is used -->
625      <xsl:when
626        test="$packages/db:seglistitem
627            [not(@userlevel=current-grouping-key())]
628            [contains(@remap,current-group()[1]/db:seg)]">
629        <xsl:text>%% </xsl:text>
630        <xsl:value-of select="current-group()[1]/db:seg"/>
631        <xsl:text> (</xsl:text>
632        <xsl:value-of select="$maxlevel"/>
633        <xsl:text>) omitted in favour of </xsl:text>
634        <xsl:value-of
635          select="$packages/db:seglistitem
636              [not(@userlevel=current-grouping-key())]
637              [contains(@remap,current-group()[1]/db:seg)]
638              /db:seg"/>
639        <xsl:text>&#xa;</xsl:text>
640      <xsl:message>
641        <xsl:if test="$maxlevel&lt;100">
642          <xsl:text> </xsl:text>
643          <xsl:if test="$maxlevel&lt;10">
644            <xsl:text> </xsl:text>
645          </xsl:if>

```

```

646         </xsl:if>
647         <xsl:value-of select="$maxlevel"/>
648         <xsl:text>. Package '</xsl:text>
649         <xsl:value-of
650             select="current-group()"/>
651         <xsl:text>' (</xsl:text>
652         <xsl:value-of
653             select="count(current-group())"/>
654         <xsl:text>*) omitted: transcluded by </xsl:text>
655         <xsl:value-of
656             select="$packages/db:seglistitem
657                 [contains(@remap,current-group()[1]/db:seg)]
658                 /db:seg"/>
659     </xsl:message>
660 </xsl:when>

```

Where the package has been flagged in the @conformance attribute, omit it.

```

661     <!-- OMIT packages flagged in conformance
662           (eg nosc, unclear legacy) -->
663     <!--
664     <xsl:when
665         test="$packages/db:seglistitem
666             [db:seg=current()/@conformance]">
667         <xsl:message>
668             <xsl:value-of select="$maxlevel"/>
669             <xsl:text>. Package '</xsl:text>
670             <xsl:value-of
671                 select="current-grouping-key()"/>
672             <xsl:text>' (</xsl:text>
673             <xsl:value-of
674                 select="count(current-group())"/>
675             <xsl:text>*) omit: </xsl:text>
676             <xsl:text>(conformance="</xsl:text>
677             <xsl:value-of
678                 select="$packages/db:seglistitem
679                     /db:seg"/>
680             <xsl:text>")</xsl:text>
681         </xsl:message>
682     </xsl:when>
683     -->
684     <!-- OTHERWISE USE THE PACKAGE -->
685     <xsl:otherwise>
686         <xsl:message>
687             <xsl:if test="number($maxlevel)<100">
688                 <xsl:text> </xsl:text>
689                 <xsl:if test="number($maxlevel)<10">
690                     <xsl:text> </xsl:text>
691                 </xsl:if>
692             </xsl:if>
693             <xsl:value-of select="$maxlevel"/>
694             <xsl:text>. Package '</xsl:text>
695             <xsl:value-of
696                 select="current-group()[1]/db:seg"/>

```

```

697         <xsl:text>' (</xsl:text>
698         <xsl:value-of
699           select="count(current-group())"/>
700         <xsl:text>*) being included</xsl:text>
701         <xsl:if
702           test="current-group()[1]/db:seg='inputenc'">
703         <xsl:text>:</xsl:text>
704         <xsl:for-each
705           select="current-group()[@userlevel=$maxlevel][1]">
706         <xsl:text> </xsl:text>
707         <xsl:value-of select="position()"/>
708         <xsl:text>)</xsl:text>
709         <xsl:value-of select="name()"/>
710         <xsl:text></xsl:text>
711         <xsl:value-of select="db:seg/@role"/>
712         <xsl:text>:</xsl:text>
713         <xsl:value-of select="$maxlevel"/>
714         </xsl:for-each>
715         </xsl:if>
716       </xsl:message>
717       <!-- ADD THE PACKAGE -->
718       <xsl:call-template name="packages">
719         <!-- for multiply-recurrent packages, ***FIX THIS ***
720           use the one with the highest userlevel -->
721         <xsl:with-param name="pkg"
722           select="current-group()[@userlevel=$maxlevel][1]">
723         <xsl:with-param name="dest" select="'doc'">
724         <!-- no cmdblocks on doc packages yet -->
725       </xsl:call-template>
726     </xsl:otherwise>
727   </xsl:choose>
728 </xsl:for-each-group>
729 </xsl:if>

```

11.3.6.2 Load the current package itself if needed : If the class or package requires itself (perhaps for demonstration or examples) this must be flagged in the <book> root element of the *ClassPack* document by including the @xlink:role attribute, set to null (or to the value of options, if required).

```

730       <!-- INCLUDE THE PACKAGE BEING CREATED? -->
731       <xsl:if test="$doctype='package'
732         and
733         /db:book/@xlink:role">
734       <xsl:message>
735         <xsl:text>Adding </xsl:text>
736         <xsl:value-of select="$name"/>
737         <xsl:text> as specified</xsl:text>
738       </xsl:message>
739       <xsl:text>\usepackage</xsl:text>
740       <xsl:if test="/db:book/@xlink:role!=''">
741         <xsl:text>[</xsl:text>
742         <xsl:value-of select="/db:book/@xlink:role"/>
743         <xsl:text>]</xsl:text>

```

```

744         </xsl:if>
745         <xsl:text>{</xsl:text>
746         <xsl:value-of select="$name"/>
747         <xsl:text>}}</xsl:text>
748         <!-- use latest rev date as the distro date -->
749         <xsl:value-of select="translate($date,'-','/')"/>
750         <xsl:text>]% added by spec&#xa;</xsl:text>
751     </xsl:if>
752     <!-- End of autopackage -->

```

11.3.7 Additional setup commands

11.3.7.1 From the user document : *ClassPack* allows authors to add setup commands as shown in [section 5.5 on page 55](#). These are `<cmdsynopsis>` elements at the end of the `<constraintdef>` for "docpackages" — this is where they get added to the .dtx file.

```

753     <xsl:apply-templates
754     select="//db:book/db:info/db:cover
755             /db:constraintdef[@xml:id='docpackages']
756             /db:cmdsynopsis[.='']"/>

```

11.3.7.2 Include the bibliography file : Add the name of the .bib file for biblatex, either from the `@xlink:href` attribute on the `<bibliography>` element, or using the name of the package.

```

757     <xsl:if test="//db:bibliography[@arch='biblatex']">
758         <xsl:text>\addbibresource{</xsl:text>
759         <xsl:choose>
760             <xsl:when
761                 test="//db:bibliography[@arch='biblatex']
762                               [@xlink:href]">
763                 <xsl:value-of
764                 select="//db:bibliography[@arch='biblatex']
765                               /@xlink:href"/>
766             </xsl:when>
767             <xsl:otherwise>
768                 <xsl:value-of select="$docname"/>
769                 <xsl:text>.bib</xsl:text>
770             </xsl:otherwise>
771         </xsl:choose>
772         <xsl:text>}&#xa;</xsl:text>
773     </xsl:if>

```

11.3.7.3 Extras from the prepost.xml file : Documentation start-up commands to be executed *before* `\begin{document}` (ie during the Preamble of the .dtx file) are taken from the prepost.xml file. The PDF metadata is also inserted here, taking its values from the document, so that it gets processed before the remaining DocTeX settings and the `\begin{document}`.

```

774     <xsl:text>\hypersetup{pdfauthor={</xsl:text>
775     <xsl:value-of

```

```

776         select="normalize-space(/db:book/db:info
777             /db:author/db:personname/db:firstname)"/>
778     <xsl:text> </xsl:text>
779     <xsl:value-of
780         select="normalize-space(/db:book/db:info
781             /db:author/db:personname/db:surname)"/>
782     <xsl:text>},
783     pdftitle={The </xsl:text>
784     <xsl:value-of select="$docname"/>
785     <xsl:text> LaTeX2e document </xsl:text>
786     <xsl:value-of select="$doctype"/>
787     <xsl:text>},
788     pdfsubject={</xsl:text>
789     <xsl:value-of
790         select="normalize-space(/db:book/db:info
791             /db:title)"/>
792     <xsl:text>},
793     pdfkeywords={</xsl:text>
794     <xsl:for-each
795         select="/db:book/db:info
796             /db:keywordset/db:keyword">
797         <xsl:if test="preceding-sibling::db:keyword">
798             <xsl:text>,</xsl:text>
799         </xsl:if>
800         <xsl:value-of select="."/>
801     </xsl:for-each>
802     <xsl:text>},
803     pdfproducer={XeLaTeX with hyperref},
804     pdfcreator={Emacs/psgml, ClassPack/Saxon, LaTeX/TeX Live}}
805 </xsl:text>
806     <xsl:for-each
807         select="/db:book/db:info/db:cover/db:constraintdef
808             [@xml:id='atendpreamble']
809             /db:procedure/db:step
810             [normalize-space(.)!='']
811             /db:cmdsynopsis/db:command">
812         <xsl:value-of select="normalize-space(.)"/>
813         <xsl:text>&#xa;</xsl:text>
814     </xsl:for-each>
815     <xsl:text>%&#xa;</xsl:text>

```

11.3.7.4 DocStrip and L^AT_EXdoc settings :

```

816     <xsl:text>%%
817     %% Settings for docstrip and ltxdoc
818     %%
819     \EnableCrossrefs
820     \CodelineIndex
821     \RecordChanges&#xa;</xsl:text>

```

11.3.7.5 The `\documentclass` command : Any stray commands that cannot conveniently be stored in the proper places but which need to be emitted *imme-*

diately after the `\begin{document}` command are taken from the `@annotations` attribute on the `<book>` root element.

```

822      <xsl:text>\begin{document}</xsl:text>
823      <xsl:value-of
824        select="normalize-space(db:book/@annotations)"/>
825      <xsl:text>&#xa; \DocInput{</xsl:text>
826      <xsl:value-of select="$docname"/>
827      <xsl:text>.dtx}&#xa;\end{document}&#xa;</xsl:text>

```

11.3.7.6 Character-set controls : After closing the driver tag that was opened in [section 11.3.5.3 on page 103](#), the checksum is computed and checked, and the character table used as a check on file corruption.

```

828      <xsl:text>%&lt;/driver>
829 % \fi
830 %
831 % \Checksum{</xsl:text>
832      <xsl:value-of select="/db:book/@security"/>
833      <xsl:text>}
834 %
835 % \CharacterTable
836 % {Upper-case  \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z
837 %   Lower-case  \a\b\c\d\e\f\g|h|i\j\k\l|m\n\o\p\q\r\s\t\u\v\w\x\y\z
838 %   Digits      \0\1\2\3\4\5\6\7\8\9
839 %   Exclamation \! Double quote \" Hash (number) \#
840 %   Dollar      \$ Percent      \% Ampersand    \&
841 %   Acute accent \' Left paren  \( Right paren  \)
842 %   Asterisk    \* Plus          \+ Comma        \,
843 %   Minus       \- Point         \. Solidus      \/
844 %   Colon       \: Semicolon    \; Less than   \&lt;
845 %   Equals      \= Greater than  \> Question mark \?
846 %   Commercial at \@ Left bracket \[ Backslash    \\
847 %   Right bracket \] Circumflex  \^ Underscore   \_
848 %   Grave accent ` Left brace   \{ Vertical bar \|
849 %   Right brace  \} Tilde       \~}
850 % &#xa;</xsl:text>

```

11.3.7.7 Revision history : Document changes recorded in the `<revhistory>` elements are emitted here for use in the changes section of the documentation.

```

851      <xsl:for-each
852        select="/db:book/db:info/db:revhistory
853              /db:revision">
854      <xsl:text>% \changes{v</xsl:text>
855      <xsl:value-of select="@version"/>
856      <xsl:text>}{</xsl:text>
857      <xsl:value-of
858        select="translate(db:date/@YYYY-MM-DD,
859                          '-','/')"/>
860      <xsl:text>}{</xsl:text>
861      <xsl:if test="db:revdescription/

```

```

862             (db:itemizedlist|db:orderedlist)
863             /db:title">
864     <xsl:value-of
865       select="normalize-space(
866         db:revdescription/
867         (db:itemizedlist|db:orderedlist)
868         /db:title)"/>
869     <xsl:text>: </xsl:text>
870 </xsl:if>
871 <xsl:choose>
872   <xsl:when test="db:revdescription/db:para">
873     <xsl:for-each select="db:revdescription/db:para">
874       <xsl:value-of
875         select="normalize-space(.)"/>
876       <xsl:if test="position()>1">
877         <xsl:text> ¶ </xsl:text>
878       </xsl:if>
879     </xsl:for-each>
880   </xsl:when>
881   <xsl:when
882     test="count(db:revdescription
883       /db:itemizedlist/db:listitem)=1">
884     <xsl:value-of
885       select="normalize-space(db:revdescription
886         /db:itemizedlist/db:listitem)"/>
887   </xsl:when>
888   <xsl:otherwise>
889     <xsl:for-each
890       select="db:revdescription/db:itemizedlist
891         /db:listitem">
892       <xsl:number/>
893       <xsl:text>) </xsl:text>
894       <xsl:value-of
895         select="normalize-space(db:para)"/>
896       <xsl:if
897         test="db:itemizedlist|db:orderedlist">
898         <xsl:text>: </xsl:text>
899       <xsl:for-each
900         select="(db:itemizedlist|db:orderedlist)
901           /db:listitem">
902         <xsl:number format="a"/>
903         <xsl:text>] </xsl:text>
904         <xsl:value-of
905           select="normalize-space(db:para)"/>
906         <xsl:if test="position() != last()">
907           <xsl:text>, </xsl:text>
908         </xsl:if>
909       </xsl:for-each>
910     </xsl:if>
911     <xsl:if test="position() != last()">
912       <xsl:text>; </xsl:text>
913     </xsl:if>
914   </xsl:for-each>
915 </xsl:otherwise>

```

```

916         </xsl:choose>
917         <xsl:text>.&#xa;</xsl:text>
918     </xsl:for-each>

```

11.3.7.8 Indexing setup : The very long list of indexing *exclusions* is used here to prevent Plain T_EX and common L^AT_EX commands (normally indexed) from being indexed. To this are added any commands in the documentation in the <command> elements unless they have a @role attribute. The intention is that only commands used in the actual class or package should get indexed.

```

919         <xsl:text>%
920 % \GetFileInfo{</xsl:text>
921         <xsl:value-of select="$docname"/>
922         <xsl:text>.dtx}&#xa;</xsl:text>
923         <xsl:text>%
924 % \DoNotIndex{\@,\@@par,\@beginparpenalty,\@empty}
925 % \DoNotIndex{\@flushglue,\@gobble,\@input,\@makefnmark}
926 % \DoNotIndex{\@makeother,\@maketitle,\@namedef,\@ne}
927 % \DoNotIndex{\@spaces,\@tempa,\@tempb,\@tempswafalse}
928 % \DoNotIndex{\@tempswatruer,\@thanks,\@thefnmark,\@topnum}
929 % \DoNotIndex{\@@,\@elt,\@forloop,\@fortmp,\@gtempa}
930 % \DoNotIndex{\@totalleftmargin,\",\/, \@ifundefined,\@nil}
931 % \DoNotIndex{\@verbatim,\@vobeyspaces,\|,\~, \ ,\active}
932 % \DoNotIndex{\advance,\aftergroup,\begingroup,\bgroup}
933 % \DoNotIndex{\mathcal,\csname,\def,\documentstyle}
934 % \DoNotIndex{\dospecials,\edef,\egroup,\else,\endcsname}
935 % \DoNotIndex{\endgroup,\endinput,\endtrivlist}
936 % \DoNotIndex{\expandafter,\fi,\fnsymbol,\futurelet,\gdef}
937 % \DoNotIndex{\global,\hbox,\hss,\if,\if@inlabel}
938 % \DoNotIndex{\if@tempswa,\if@twocolumn,\ifcase,\ifcat}
939 % \DoNotIndex{\iffalse,\ifx,\ignorespaces,\index,\input}
940 % \DoNotIndex{\item,\jobname,\kern,\leavevmode,\leftskip}
941 % \DoNotIndex{\let,\llap,\lower,\m@ne,\next,\newpage}
942 % \DoNotIndex{\nobreak,\noexpand,\nonfrenchspacing}
943 % \DoNotIndex{\obeylines,\or,\protect,\raggedleft}
944 % \DoNotIndex{\rightskip,\rm,\sc,\setbox,\setcounter}
945 % \DoNotIndex{\small,\space,\string,\strut,\strutbox}
946 % \DoNotIndex{\thefootnote,\thispagestyle,\topmargin}
947 % \DoNotIndex{\trivlist,\tt,\twocolumn,\typeout,\vss,\vtop}
948 % \DoNotIndex{\xdef,\z@,\,,\@bsphack,\@esphack,\@noligs}
949 % \DoNotIndex{\@vobeyspaces,\@xverbatim,\`,\catcode,\end}
950 % \DoNotIndex{\escapechar,\frenchspacing,\glossary}
951 % \DoNotIndex{\hangindent,\hfil,\hfill,\hskip,\hspace,\ht}
952 % \DoNotIndex{\it,\langle,\leaders,\long,\makelabel}
953 % \DoNotIndex{\marginpar,\markboth,\mathcode,\mathsurround}
954 % \DoNotIndex{\mbox,\newcount,\newdimen,\newskip}
955 % \DoNotIndex{\nopagebreak,\parfillskip,\parindent}
956 % \DoNotIndex{\parskip,\penalty,\raise,\rangle,\section}
957 % \DoNotIndex{\setlength,\TeX,\topsep,\underline,\unskip}
958 % \DoNotIndex{\verb,\vskip,\vspace,\widetilde,\,,\%, \@date}
959 % \DoNotIndex{\@defpar,\[, \{, \}, \], \count@, \ifnum, \loop}
960 % \DoNotIndex{\today,\uppercase,\uccode,\baselineskip}
961 % \DoNotIndex{\begin,\tw@,\a,\b,\c,\d,\e,\f,\g,\h,\i,\j,\k}

```

```

962 % \DoNotIndex{\l,\m,\n,\o,\p,\q,\r,\s,\t,\u,\v,\w,\x,\y,\z}
963 % \DoNotIndex{\A,\B,\C,\D,\E,\F,\G,\H,\I,\J,\K,\L,\M,\N,\O}
964 % \DoNotIndex{\P,\Q,\R,\S,\T,\U,\V,\W,\X,\Y,\Z,\1,\2,\3,\4}
965 % \DoNotIndex{\5,\6,\7,\8,\9,\0,\!,\#\,\$,\\&,\',\(\,\)}
966 % \DoNotIndex{\+,\\. ,\:,\\;,\\&lt;,\=,\\>,\?,\\_,\\discretionary}
967 % \DoNotIndex{\\immediate,\\makeatletter,\\makeatother}
968 % \DoNotIndex{\\meaning,\\newenvironment,\\par,\\relax}
969 % \DoNotIndex{\\renewenvironment,\\repeat,\\scriptsize}
970 % \DoNotIndex{\\selectfont,\\the,\\undefined,\\arabic,\\do}
971 % \DoNotIndex{\\makeindex,\\null,\\number,\\show,\\write,\\@ehc}
972 % \DoNotIndex{\\@author,\\@ehc,\\@ifstar,\\@sanitize,\\@title}
973 % \DoNotIndex{\\everypar,\\if@minipage,\\if@restonecol,\\ifeof}
974 % \DoNotIndex{\\ifmmode,\\lccode,\\newtoks,\\onecolumn,\\openin}
975 % \DoNotIndex{\\p@,\\SelfDocumenting,\\settowidth}
976 % \DoNotIndex{\\@resetonecoltrue,\\@resetonecolfalse,\\bf}
977 % \DoNotIndex{\\clearpage,\\closein,\\lowercase,\\@tempdima}
978 % \DoNotIndex{\\@inlabelfalse,\\selectfont,\\mathcode}
979 % \DoNotIndex{\\newmathalphabet,\\rmdefault,\\bfdefault}
980 % \DoNotIndex{\\DeclareRobustCommand,\\@ifpackagewith}
981 % \DoNotIndex{\\#,\\%,\\&,\\*,\\-,\\^,\\_,\\|,\\~,\\$}
982 % \DoNotIndex{\\acro,\\addbibresource,\\addcontentsline,\\addtolength}
983 % \DoNotIndex{\\allowbreak,\\alph,\\@Alph,\\and,\\appendix,\\arrayrulewidth}
984 % \DoNotIndex{\\ast,\\baselinestretch,\\bfseries,\\bgroup,\\Bib,\\BibTeX}
985 % \DoNotIndex{\\BiBTeX,\\BIBTeX,\\bigskip,\\box,\\caption,\\centering,\\char}
986 % \DoNotIndex{\\CharacterTable,\\Checksum,\\citeyear,\\cjktext,\\ClassError}
987 % \DoNotIndex{\\classorpackage,\\CodelineIndex,\\color,\\colorbox}
988 % \DoNotIndex{\\columnsep,\\columnwidth,\\Con,\\ConTeXt}
989 % \DoNotIndex{\\CPKannotationindent,\\CPKdocname,\\CPKdoctype,\\CPKmenusep}
990 % \DoNotIndex{\\CPKpoststrut,\\CPKprestrut,\\CPKrevmarg,\\CPKrunningecho}
991 % \DoNotIndex{\\CPKthisjob,\\CPKthispackage,\\CPKvstrut,\\c@section}
992 % \DoNotIndex{\\@currsiz,\\DeclareOption,\\declarepostamble}
993 % \DoNotIndex{\\declarepreamble,\\DescribeColor,\\DescribeEnv}
994 % \DoNotIndex{\\DescribeError,\\DescribeOption,\\DescribePackage}
995 % \DoNotIndex{\\DescribeTemplate,\\descriptionlabel,\\divide,\\DoNotIndex}
996 % \DoNotIndex{\\dotfill,\\dots,\\@dottedtocline,\\DoubleperCent,\\doxitem}
997 % \DoNotIndex{\\dp,\\egroup,\\emph,\\empty,\\EnableCrossrefs}
998 % \DoNotIndex{\\encodingdefault,\\endbatchfile,\\endpreamble,\\enspace}
999 % \DoNotIndex{\\ensuremath,\\fa,\\fbox,\\fboxrule,\\fboxsep,\\file,\\Finale}
1000 % \DoNotIndex{\\flushright,\\fnote,\\font,\\fontdimen,\\fontencoding}
1001 % \DoNotIndex{\\fontfamily,\\fontseries,\\fontshape,\\fontsize,\\footnote}
1002 % \DoNotIndex{\\footnotesize,\\from,\\generate,\\GetFileInfo,\\HandRight}
1003 % \DoNotIndex{\\@height,\\@@hline,\\href,\\hrule,\\hsize,\\huge,\\Huge}
1004 % \DoNotIndex{\\hyperref,\\hypersetup,\\hyphenation,\\ifdim,\\@ifnextchar}
1005 % \DoNotIndex{\\itshape,\\keepsilent,\\keys,\\labelenumi,\\LabelFont}
1006 % \DoNotIndex{\\labelformat,\\large,\\Large,\\LARGE,\\LaTeX,\\LaTeXe}
1007 % \DoNotIndex{\\leftmark,\\lfoot,\\lhead,\\longestline,\\lstloadlanguages}
1008 % \DoNotIndex{\\l@section,\\l@subsubsection,\\ltx@ifpackageloaded}
1009 % \DoNotIndex{\\LyX,\\MacroFont,\\marginfont,\\marginnote,\\medskip}
1010 % \DoNotIndex{\\menu,\\menusep,\\message,\\MF,\\@minus,\\MP,\\Msg}
1011 % \DoNotIndex{\\multirow,\\NeedsTeXFormat,\\newcommand,\\newcounter}
1012 % \DoNotIndex{\\newgeometry,\\newlength,\\newwrite,\\nicefrac,\\noalign}
1013 % \DoNotIndex{\\nocite,\\noindent,\\nolinkurl,\\nopreamble,\\normalfont}
1014 % \DoNotIndex{\\normalsize,\\numberstring,\\obeyspaces,\\ordinal}
1015 % \DoNotIndex{\\PackageError,\\pageref,\\phantomsection,\\@plus}

```

```

1016 % \DoNotIndex{\@pnumwidth,\preamble,\PrintChanges}
1017 % \DoNotIndex{\printexternalcurrentfont,\PrintIndex,\Provides}
1018 % \DoNotIndex{\qqquad,\quad,\raisebox,\RecordChanges,\reflectbox}
1019 % \DoNotIndex{\refname,\renewcommand,\renewmenumacro,\reserved@a}
1020 % \DoNotIndex{\rightarrow,\rightmark,\rmfamily,\rotatebox,\rule}
1021 % \DoNotIndex{\school,\sloppy,\smallskip,\SMC,\SMC@unknown@warning}
1022 % \DoNotIndex{\Square,\stanza,\star,\@startsection,\stepcounter}
1023 % \DoNotIndex{\StopEventually,\subsubsection,\tableofcontents}
1024 % \DoNotIndex{\textbackslash,\textbf,\textdegree,\texteiyad}
1025 % \DoNotIndex{\textheight,\textit,\textlangle,\textrangle,\textsf}
1026 % \DoNotIndex{\textSMC,\textsuperscript,\texttt,\TheSbox}
1027 % \DoNotIndex{\thesection,\thinspace,\tiny,\@tocrmarg}
1028 % \DoNotIndex{\tubhideheight,\tubreflect,\uline,\updefault}
1029 % \DoNotIndex{\upshape,\use@babel,\@usebib,\usedir,\usepostamble}
1030 % \DoNotIndex{\usepreamble,\vbox,\vfill,\vrefrange,\vrule}
1031 % \DoNotIndex{\Xe,\XeLaTeX,\XeTeX,\@xhline}
1032 </xsl:text>
1033 <!-- add entry for examples unless with spaces -->
1034 <xsl:for-each-group
1035   select="//db:command
1036     [not(@role)]
1037     [not(contains(.,' ') or contains(.,'{')]]"
1038   group-by="normalize-space(.)">
1039   <xsl:text>% \DoNotIndex{</xsl:text>
1040   <xsl:value-of select="current-grouping-key()"/>
1041   <xsl:text>}&#xa;</xsl:text>
1042 </xsl:for-each-group>

```

11.3.7.9 Deferred commands from the user document : Lastly, any startup commands which need to be emitted after the `\begin{document}` command are output here, shielded with `\makeatletter` if needed.

```

1043 <!-- Documentation start-up commands to execute
1044   *after* \begin{document} (ie during the
1045   absorption phase of the .dtx file running
1046   from \DocInput) -->
1047 <xsl:for-each
1048   select="//db:book/db:info/db:cover
1049     /db:constraintdef
1050     [@xml:id='atbeginndoc']
1051     /db:procedure/db:step
1052     [normalize-space(.)!='']
1053     /db:cmdsynopsis/db:command">
1054   <xsl:if test="contains(.,'@')">
1055     <xsl:text>% \makeatletter&#xa;</xsl:text>
1056   </xsl:if>
1057   <xsl:text>% </xsl:text>
1058   <xsl:value-of select="normalize-space(.)"/>
1059   <xsl:text>&#xa;</xsl:text>
1060   <xsl:if test="contains(.,'@')">
1061     <xsl:text>% \makeatother&#xa;</xsl:text>
1062   </xsl:if>
1063 </xsl:for-each>

```

```
1064      <xsl:text>%&#xa;</xsl:text>
```

11.3.8 Process the document parts

```
1065      <!-- now the fun starts -->
1066      <xsl:apply-templates
1067        select="db:book/db:info |
1068              db:book/db:part[@xml:id='doc'] |
1069              db:book/db:part[@xml:id='code'] |
1070              db:book/db:part[@xml:id='files']"/>
1071      <xsl:text>&#xa;</xsl:text>
1072    </xsl:otherwise>
1073  </xsl:choose>
1074 </xsl:template>
1075 <!-- end of root templae -->
```

11.4 Pattern templates

11.4.1 Metadata

db:abstract Emit the Abstract as standard L^AT_EX, changing the title if given, and resetting the paragraph gap and indent if the parskip package has been used.

```
1076  <xsl:template match="db:info/db:abstract[@role='CTAN']"/>
1077
1078  <xsl:template match="db:legalnotice"/>
1079
1080  <xsl:template match="db:info/db:abstract[not(@role='CTAN')]"/>
1081    <xsl:if test="db:title">
1082      <xsl:text>% \renewcommand{\abstractname}{</xsl:text>
1083      <xsl:apply-templates select="db:title/node()"/>
1084      <xsl:text>}\thispagestyle{empty}&#xa;</xsl:text>
1085    </xsl:if>
1086    <xsl:text>% \begin{abstract}&#xa;</xsl:text>
1087    <xsl:if
1088      test="not(//db:constraintdef[@xml:id='docpackages']//
1089        db:seglistitem/db:seg[.='parskip']
1090        [@condition='off'])">
1091      <xsl:text>% \parskip=0.5\baselineskip
1092      % \advance\parskip by 0pt plus 2pt
1093      % \parindent=0pt</xsl:text>
1094    </xsl:if>
1095    <xsl:text>% \noindent&#xa;</xsl:text>
1096    <xsl:apply-templates/>
1097    <xsl:if test="@xlink:href and not(@xlink:show='none')">
1098      <xsl:text>% \par\centering
1099      % \includegraphics[width=.666\columnwidth]{</xsl:text>
1100      <xsl:value-of select="@xlink:href"/>
1101      <xsl:text>}&#xa;</xsl:text>
1102    </xsl:if>
1103    <xsl:text>% \end{abstract}
1104    % \clearpage
1105    % \tableofcontents&#xa;</xsl:text>
1106    <xsl:if test="//db:exceptionname">
1107      <xsl:apply-templates
```

```

1108         select="document('rfc2119.xml')/db:section"/>
1109     </xsl:if>
1110     <xsl:apply-templates
1111         select="document('representation.xml')/db:section"/>
1112 </xsl:template>

```

After the Abstract, see if there are any uses of <exceptionname>, and if so, emit the warning text from rfc2119.xml.

db:cmdsynopsis The <cmdsynopsis> element holds any special user definitions for the documentation, called in [section 11.3.7.1 on page 113](#).

This usage is deprecated as of version 1.18 of *ClassPack*. Whole definition commands SHOULD now go in the code verbatim.

Start this block with \makeatletter if the command contains an ‘at’ sign (@), then proceed by category.

```

1113 <xsl:template match="db:cmdsynopsis">
1114     <xsl:if test="position()=1
1115                 and
1116                 ../db:cmdsynopsis[contains(., '@')]">
1117         <xsl:text>\makeatletter&#xa;</xsl:text>
1118     </xsl:if>

```

Commands to be emitted as they stand.

```

1119 <xsl:choose>
1120     <xsl:when test="db:command/@role='asis'">
1121         <xsl:value-of select="db:command"/>
1122         <xsl:text>&#xa;</xsl:text>
1123     </xsl:when>

```

New output files.

```

1124 <!-- new \write files: new: no arg -->
1125 <xsl:when test="db:command/@remap='write'">
1126     <xsl:text>\newwrite</xsl:text>
1127     <xsl:value-of select="db:command"/>
1128     <xsl:text>&#xa;</xsl:text>
1129 </xsl:when>

```

New counters.

```

1130 <!-- new counters -->
1131 <xsl:when test="db:command/@remap='counter'">
1132     <xsl:text>\newcounter{</xsl:text>
1133     <xsl:value-of select="db:command"/>
1134     <xsl:text>}</xsl:text>
1135     <xsl:if test="db:arg">
1136         <xsl:text>\setcounter{</xsl:text>
1137         <xsl:value-of select="db:command"/>
1138         <xsl:text>}</xsl:text>

```

```

1139         <xsl:value-of select="db:arg"/>
1140         <xsl:text>}</xsl:text>
1141     </xsl:if>
1142     <xsl:text>&#xa;</xsl:text>
1143 </xsl:when>

```

New lengths.

```

1144 <!-- new lengths -->
1145 <xsl:when test="db:command/@remap='length'">
1146     <xsl:text>\newlength{\</xsl:text>
1147     <xsl:value-of select="db:command"/>
1148     <xsl:text>}</xsl:text>
1149     <xsl:if test="db:arg">
1150         <xsl:text>\setlength{\</xsl:text>
1151         <xsl:value-of select="db:command"/>
1152         <xsl:text>}{</xsl:text>
1153         <xsl:value-of select="db:arg"/>
1154         <xsl:text>}</xsl:text>
1155     </xsl:if>
1156     <xsl:text>&#xa;</xsl:text>
1157 </xsl:when>

```

Plain TeX commands, possibly with arguments.

```

1158 <!-- plain TeX commands -->
1159 <xsl:when test="@language='TeX'">
1160     <xsl:if test="@role='long'">
1161         <xsl:text>\long</xsl:text>
1162     </xsl:if>
1163     <xsl:text>\def\</xsl:text>
1164     <xsl:value-of select="db:command"/>
1165     <xsl:if test="db:arg/@wordsize">
1166         <xsl:call-template name="repeatarg">
1167             <xsl:with-param name="limit"
1168                 select="db:arg/@wordsize"/>
1169             <xsl:with-param name="prefix"
1170                 select="substring(db:arg/@annotations,1,
1171                     (string-length(db:arg/@annotations)
1172                     div 2))"/>
1173             <xsl:with-param name="suffix"
1174                 select="substring(db:arg/@annotations,
1175                     (string-length(db:arg/@annotations)
1176                     div 2)+1)"/>
1177         </xsl:call-template>
1178     </xsl:if>
1179     <xsl:choose>
1180         <xsl:when test="db:arg/@remap">
1181             <xsl:text>{</xsl:text>
1182             <xsl:value-of
1183                 select="/db:book/@*[name()='current()'
1184                     /db:arg/@remap]"/>
1185             <xsl:text>}&#xa;</xsl:text>
1186         </xsl:when>
1187         <xsl:otherwise>

```

```

1188         <xsl:text>{%&#xa;&#9;</xsl:text>
1189         <xsl:value-of select="db:arg"/>
1190         <xsl:text>}&#xa;</xsl:text>
1191     </xsl:otherwise>
1192 </xsl:choose>
1193 </xsl:when>

```

New or renewed L^AT_EX environments.

```

1194     <xsl:when test="db:command/@remap='environment'
1195                 and count(db:command/db:arg)=2">
1196         <xsl:choose>
1197             <xsl:when test="db:command/@role='renew'">
1198                 <xsl:text>\renewenvironment{</xsl:text>
1199             </xsl:when>
1200             <xsl:otherwise>
1201                 <xsl:text>\newenvironment{</xsl:text>
1202             </xsl:otherwise>
1203         </xsl:choose>
1204         <xsl:value-of select="db:command"/>
1205         <xsl:text>}</xsl:text>
1206         <!-- number of arguments, if any -->
1207         <xsl:if test="db:arg[1]/@wordsize">
1208             <xsl:text>[</xsl:text>
1209             <xsl:value-of select="db:arg[1]/@wordsize"/>
1210             <xsl:text>]</xsl:text>
1211         </xsl:if>
1212         <xsl:text>{%&#xa;&#9;</xsl:text>
1213         <xsl:value-of select="db:arg[1]"/>
1214         <xsl:text>}&#xa;</xsl:text>
1215         <xsl:text>{%&#xa;&#9;</xsl:text>
1216         <xsl:value-of select="db:arg[2]"/>
1217         <xsl:text>}&#xa;</xsl:text>
1218     </xsl:when>

```

Anything else is considered a new or renewed L^AT_EX command.

```

1219     <!-- LaTeX commands -->
1220     <xsl:otherwise>
1221         <xsl:choose>
1222             <xsl:when test="db:command/@role='renew'">
1223                 <xsl:text>\renewcommand{</xsl:text>
1224             </xsl:when>
1225             <xsl:otherwise>
1226                 <xsl:text>\newcommand{</xsl:text>
1227             </xsl:otherwise>
1228         </xsl:choose>
1229         <xsl:value-of select="db:command"/>
1230         <xsl:text>}</xsl:text>
1231         <!-- number of arguments, if any -->
1232         <xsl:if test="db:arg/@wordsize">
1233             <xsl:text>[</xsl:text>
1234             <xsl:value-of select="db:arg/@wordsize"/>
1235             <xsl:text>]</xsl:text>
1236         <xsl:if test="db:arg/@condition">

```

```

1237         <xsl:text>[</xsl:text>
1238         <xsl:value-of select="db:arg/@condition"/>
1239         <xsl:text>]</xsl:text>
1240     </xsl:if>
1241 </xsl:if>
1242 <xsl:choose>
1243     <!-- remap is a reference to an attribute of the
1244          root element, eg arch means use @arch -->
1245     <xsl:when test="db:arg/@remap">
1246         <xsl:text>[</xsl:text>
1247         <xsl:value-of
1248             select="/db:book/@*[name()=
1249                 current()/db:arg/@remap]"/>
1250         <xsl:text>]&#xa;</xsl:text>
1251     </xsl:when>
1252     <xsl:otherwise>
1253         <xsl:text>[%&#xa;&#9;</xsl:text>
1254         <xsl:value-of select="db:arg"/>
1255         <xsl:text>]&#xa;</xsl:text>
1256     </xsl:otherwise>
1257 </xsl:choose>
1258 </xsl:otherwise>

```

End the block with `\makeatother` if needed.

```

1259 </xsl:choose>
1260 <xsl:if test="position()=last()
1261             and
1262             ../db:cmdsynopsis[contains(., '@')]">
1263     <xsl:text>\makeatother&#xa;</xsl:text>
1264 </xsl:if>
1265 </xsl:template>

```

`db:info` Issue notification of the document type and version, then process the contents.

```

1266 <xsl:template match="db:info">
1267     <xsl:message>
1268         <xsl:text>Creating </xsl:text>
1269         <xsl:value-of select="$doctype"/>
1270         <xsl:text> ' </xsl:text>
1271         <xsl:value-of select="$docname"/>
1272         <xsl:text>' (v</xsl:text>
1273         <xsl:value-of select="$version"/>
1274         <xsl:text>.</xsl:text>
1275         <xsl:value-of select="$revision"/>
1276         <xsl:text>) with checksum </xsl:text>
1277         <xsl:value-of select="/db:book/@security"/>
1278         <xsl:text>.</xsl:text>
1279     </xsl:message>
1280     <xsl:apply-templates/>
1281 </xsl:template>
1282
1283 <xsl:template match="db:cover"/>

```

We omit the the cover settings, which were done earlier as part of the titling.

db:title [in `db:info/` and `db:info/db:subtitle`]

For the metadata title (title of the class or package), if this is a class, or a package which has not been preloaded for the documentation, ensure the `\fileversion` and `\filedate` values are made available.

```

1284 <xsl:template match="db:info/db:title">
1285   <xsl:if test="/db:book/@arch='package' or
1286             not(/db:book/@xlink:role)">
1287     <xsl:text>% \def\fileversion{</xsl:text>
1288     <xsl:value-of select="/db:book/@version"/>
1289     <xsl:text>.</xsl:text>
1290     <xsl:value-of select="/db:book/@revision"/>
1291     <xsl:text>}&#xa;% \def\filedate{</xsl:text>
1292     <xsl:value-of
1293       select="translate(/db:book/db:info
1294                        /db:revhistory/db:revision
1295                        [@version=max(parent::db:revhistory
1296                                   /db:revision/@version)]
1297                        /db:date/@YYYY-MM-DD,'-','/')">
1298     <xsl:text>}&#xa;</xsl:text>
1299   </xsl:if>

```

db:title **Make the title itself**

Then construct the formal `\title`, with the name of the document type.

```

1300 <xsl:text>% \title{The </xsl:text>
1301 <xsl:text> \textsf{</xsl:text>
1302 <xsl:value-of select="$docname"/>
1303 <xsl:text>} </xsl:text>
1304 <xsl:choose>
1305   <xsl:when test="/db:book/@arch='script'">
1306     <xsl:text></xsl:text>
1307   </xsl:when>
1308   <xsl:otherwise>
1309     <xsl:text>\LaTeXe\ </xsl:text>
1310   </xsl:otherwise>
1311 </xsl:choose>
1312 <xsl:choose>
1313   <xsl:when test="$doctype='class'">
1314     <xsl:text>document class</xsl:text>
1315     <xsl:if
1316       test="count(/db:part//db:appendix
1317                  [substring-after(@xlink:href,'.')
1318                  = 'cls'])>1">
1319       <xsl:text>es</xsl:text>
1320     </xsl:if>
1321   </xsl:when>
1322   <xsl:otherwise>
1323     <xsl:value-of select="$doctype"/>
1324   </xsl:otherwise>
1325 </xsl:choose>

```

db:title Footnotes for title page

Add a \thanks footnote for the version

```

1326     <xsl:text>\thanks{%
1327 % This document corresponds to </xsl:text>
1328     <xsl:text>\textsf{</xsl:text>
1329     <xsl:value-of select="$docname"/>
1330     <xsl:text>}&#xa;% \textit{v.}\ \fileversion </xsl:text>
1331     <xsl:choose>
1332         <xsl:when test="/db:book/@status='alpha' or
1333             /db:book/@status='beta'">
1334             <xsl:text>$\</xsl:text>
1335             <xsl:value-of select="/db:book/@status"/>
1336             <xsl:text>$\</xsl:text>
1337         </xsl:when>
1338         <!-- alpha, beta, delta (draft), phi (final),
1339             or pi (production) -->
1340         <xsl:when test="/db:book/@status='α' or
1341             /db:book/@status='β' or
1342             /db:book/@status='δ' or
1343             /db:book/@status='φ' or
1344             /db:book/@status='π'">
1345             <xsl:value-of select="/db:book/@status"/>
1346         </xsl:when>
1347         <xsl:when test="/db:book/@status">
1348             <xsl:value-of select="/db:book/@status"/>
1349         </xsl:when>
1350     </xsl:choose>
1351     <xsl:text>, dated \filedate.</xsl:text>
1352     <!-- add second note for sponsors -->
1353     <xsl:if test="ancestor::db:info/descendant::db:author/
1354         db:contrib[@role='sponsor']">
1355         <xsl:text>\enspace\thanks{%
1356 % Supported by </xsl:text>
1357         <xsl:for-each
1358             select="ancestor::db:info/descendant::db:author/
1359                 db:contrib[@role='sponsor']">
1360             <xsl:if test="position()>1">
1361                 <xsl:text>,</xsl:text>
1362                 <xsl:if test="position()=last()">
1363                     <xsl:text>and </xsl:text>
1364                 </xsl:if>
1365             </xsl:if>
1366             <xsl:apply-templates/>
1367         </xsl:for-each>
1368         <xsl:text>.</xsl:text>
1369     </xsl:if>

```

Use the contents of the <title> element as the explanatory subtitle (and any <subtitle> as a second line; exclude the <subtitle> from additional processing).

```

1370     <xsl:text>&#xa;% \[1em]\Large &#xa;% </xsl:text>
1371     <xsl:apply-templates/>

```

```

1372     <xsl:if test="following-sibling::db:subtitle
1373                 [not(@role='labelonly')]"]>
1374         <xsl:text>&#xa;% \[1ex]\large &#xa;% </xsl:text>
1375         <xsl:apply-templates
1376             select="following-sibling::db:subtitle/node()"/>
1377     </xsl:if>
1378     <xsl:text>}&#xa;</xsl:text>
1379 </xsl:template>
1380
1381 <xsl:template match="db:info/db:subtitle"/>
1382 <xsl:template match="db:info/db:titleabbrev"/>
1383
1384 <xsl:variable name="shorttitle">
1385     <xsl:choose>
1386         <xsl:when test="/db:book/db:info/db:titleabbrev">
1387             <xsl:value-of select="/db:book/db:info/db:titleabbrev/node()"/>
1388         </xsl:when>
1389         <xsl:otherwise>
1390             <xsl:value-of select="/db:book/db:info/db:title/node()"/>
1391         </xsl:otherwise>
1392     </xsl:choose>
1393 </xsl:variable>

```

db:author [in db:info//]

Output the \author before the first author, process the names, add titles and/or affiliations, then repeat for next author, and output a \maketitle after the last one.

```

1394 <xsl:template match="db:info//db:author">
1395     <xsl:if test="count(preceding-sibling::db:author)=0">
1396         <xsl:text>% \author{</xsl:text>
1397     </xsl:if>
1398     <xsl:for-each select="db:personname/db:*">
1399         <xsl:apply-templates/>
1400         <xsl:if test="position()=last()">
1401             <xsl:text> </xsl:text>
1402         </xsl:if>
1403     </xsl:for-each>
1404     <xsl:apply-templates
1405         select="db:honorific | db:affiliation"/>
1406     <xsl:if test="db:email">
1407         <xsl:text>\\normalsize{\url{</xsl:text>
1408         <xsl:value-of select="db:email"/>
1409         <xsl:text>}}</xsl:text>
1410     </xsl:if>
1411     <xsl:choose>
1412         <xsl:when test="following-sibling::db:author">
1413             <xsl:text>&#xa;% \and&#xa;% </xsl:text>
1414         </xsl:when>
1415         <xsl:otherwise>
1416             <xsl:text>}&#xa;% \maketitle&#xa;</xsl:text>
1417         </xsl:otherwise>
1418     </xsl:choose>

```

```
1419 </xsl:template>
```

db:honorific Capitalise honorifics (eg degrees).

```
1420 <xsl:template match="db:honorific">
1421   <xsl:text> \textsc{</xsl:text>
1422   <xsl:value-of
1423     select="translate(.,
1424       'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
1425       'abcdefghijklmnopqrstuvwxyz')"/>
1426   <xsl:text>}</xsl:text>
1427 </xsl:template>
```

db:affiliation Capitalise affiliations likewise, on separate lines.

```
1428 <xsl:template match="db:affiliation">
1429   <xsl:text>\\normalsize </xsl:text>
1430   <xsl:for-each select="db:*">
1431     <xsl:value-of select="normalize-space(.)"/>
1432     <!-- capitalisation for Centre, Hospital, Unit,
1433       Project, and maybe others -->
1434     <xsl:if test="@remap">
1435       <xsl:text> </xsl:text>
1436       <xsl:value-of
1437         select="translate(substring(@remap,1,1),
1438           'cuhp','CUHP')"/>
1439       <xsl:value-of select="substring(@remap,2)"/>
1440     </xsl:if>
1441     <xsl:if test="position()=last()">
1442       <xsl:text>\\[-.25ex]normalsize </xsl:text>
1443     </xsl:if>
1444   </xsl:for-each>
1445 </xsl:template>
```

db:releaseinfo [in db:info/ and db:info/db:annotation and db:info/db:revhistory and db:info/db:copyright]

Omit elements handled elsewhere.

```
1446 <xsl:template match="db:info/db:releaseinfo |
1447   db:info/db:annotation |
1448   db:info/db:revhistory |
1449   db:info/db:copyright"/>
```

11.4.2 Hierarchy

db:preface Normal prefaces have a title and get a ToC entry. With no title they are treated informally and centered (eg for ‘thanks’ sections). An @xml:id will be used as a label, if present.

```
1450 <xsl:template match="db:preface">
1451   <xsl:choose>
1452     <xsl:when test="db:title">
```

```

1453      <xsl:text>% \clearpage\section*{</xsl:text>
1454      <xsl:apply-templates select="db:title/node()"/>
1455      <xsl:text>}\addcontentsline{toc}</xsl:text>
1456      <xsl:text>{subsection}{</xsl:text>
1457      <xsl:apply-templates select="db:title/node()"/>
1458      <xsl:text>}</xsl:text>
1459    </xsl:when>
1460    <xsl:otherwise>
1461      <xsl:text>% \clearpage\null\vfill</xsl:text>
1462      <xsl:text>\begingroup\centering</xsl:text>
1463    </xsl:otherwise>
1464  </xsl:choose>
1465  <xsl:if test="@xml:id">
1466    <xsl:text>\label{</xsl:text>
1467    <xsl:value-of select="@xml:id"/>
1468    <xsl:text>}</xsl:text>
1469  </xsl:if>
1470  <xsl:text>&#xa;</xsl:text>
1471  <xsl:apply-templates/>
1472  <xsl:if test="not(db:title)">
1473    <xsl:text>% \par\endgroup\vfill&#xa;</xsl:text>
1474  </xsl:if>
1475 </xsl:template>

```

db:part There are three possible parts, identified in their @xml:id attribute as "doc", "code", and "files", as explained in [section 3.2 on page 20](#).

```

1476   <xsl:template match="db:part">
1477     <xsl:choose>

```

db:part Documentation

For the documentation, output a 'Latest changes' section first, before the content, and add the bibliography at the end if needed.

```

1478   <xsl:when test="@xml:id='doc'">
1479     <xsl:text>% \clearpage&#xa;</xsl:text>
1480     <xsl:text>% \section*{Latest changes}&#xa;</xsl:text>
1481     <xsl:apply-templates
1482       select="//db:revhistory/db:revision
1483         [@version=$latestrevhist]"/>
1484     <xsl:for-each
1485       select="//db:revhistory/db:revision
1486         [@version!=$latestrevhist]">
1487       <xsl:sort select="db:date/@YYYY-MM-DD"
1488         order="descending"/>
1489       <xsl:if test="position()&lt;4">
1490         <xsl:apply-templates select="."/>
1491       </xsl:if>
1492     </xsl:for-each>
1493     <xsl:text>% \par\bigskip
1494 % See p.\thinspace\pageref{</xsl:text>
1495     <xsl:value-of
1496       select="//db:book/db:info/db:revhistory/@xml:id"/>

```

```

1497      <xsl:text>} for earlier changes.&#xa;</xsl:text>
1498      <xsl:apply-templates/>
1499      <xsl:if
1500        test="not(db:bibliography) and
1501              descendant::db:exceptionname and
1502              (not(@conformance)
1503                or
1504                 (@conformance!='xelatex' and
1505                  @conformance!='lualatex'))">
1506        <xsl:text>% \bibliography{</xsl:text>
1507        <xsl:value-of select="$docname"/>
1508        <xsl:text>}\bibliographystyle{apalike}%&#xa;</xsl:text>
1509      </xsl:if>
1510    </xsl:when>

```

db:part **Code**

Output the termination for the documentation, resetting the margin width and including the change history and the index.

```

1511      <xsl:when test="@xml:id='code'">
1512      <!-- the invocation of \newgeometry has been withdrawn -->
1513      <xsl:text>% \StopEventually{\label{endcode}%
1514 % \clearpage
1515 % \addcontentsline{toc}{section}{Change History}%
1516 % \label{</xsl:text>
1517      <xsl:value-of
1518        select="/db:book/db:info/db:revhistory/@xml:id"/>
1519      <xsl:text>}%
1520 % \PrintChanges
1521 % \clearpage
1522 % \label{codeindex}%
1523 % \addcontentsline{toc}{section}{Index}%
1524 % \PrintIndex}&#xa;</xsl:text>

```

db:part **Margin width calculation**

The margin width is recalculated to take account of the widest marginal macro or other tag name, outputting it as a L^AT_EX length to be set at run time. The longest length is found by iteration through commands and environment variables to be referenced, in order of length, longest first.

```

1525      <!-- Calculate margin width from longest string -->
1526      <!-- WITHDRAWN 2024-02-04
1527      <xsl:for-each-group group-by="normalize-space()"
1528        select="descendant::db:command
1529                [@role and not(@xml:lang or
1530                  contains(.,'[') or contains(.,'{')])
1531                |
1532                descendant::db:envar
1533                [@role and not(@xml:lang or
1534                  contains(.,'[') or contains(.,'{')])
1535                |
1536                descendant::db:tag[@role]
1537                |

```

```

1538             descendant::db:varname[@role]
1539             |
1540             descendant::db:classname[@role]
1541             |
1542             descendant::db:package[@role]
1543             |
1544             descendant::db:annotation/@xreflabel">
1545     <xsl:sort
1546       select="string-length(current-grouping-key())"
1547       data-type="number" order="descending"/>
1548     <xsl:if test="position()='1'">
1549       <xsl:text>% \divide\CPKrevmarg by2 &#xa;</xsl:text>
1550       <xsl:text>% \addtolength{\CPKrevmarg}{%&#xa;</xsl:text>
1551       <xsl:text>% \widthof{\LabelFont{</xsl:text>
1552       <xsl:value-of select="current-grouping-key()"/>
1553       <xsl:text>}}>
1554     % \newgeometry{left=\CPKrevmarg}&#xa;</xsl:text>
1555       <xsl:text>% \message{Margin reset to </xsl:text>
1556       <xsl:text>\the\CPKrevmarg, to fit &lt;</xsl:text>
1557       <xsl:value-of select="current-grouping-key()"/>
1558       <xsl:text>>}&#xa;</xsl:text>
1559       <xsl:text>% \makeatletter\ltx@ifpackageloaded{fancyhdr}
1560     % {\lhead{\footnotesize\rightmark}\lfoot{\footnotesize\leftmark}}
1561     % {\relax}\makeatother&#xa;</xsl:text>
1562     </xsl:if>
1563   </xsl:for-each-group>
1564   end of withdrawn margin recalc code -->

```

db:part **Code content**

The code part could be a monolith (no chapters) so it would all be one package or class. But if (usually) it has chapters, the tags for the primary output need to be set round the chapters, and any appendices will be separately-annotated ancillary files.

Prerequisites like the fix-cm package⁵ and any xcolor swatch options need to be output here, before anything else, so they get handled while we process the content of this part, starting with the chapters which are the primary content (see [section 11.4.2 on page 133](#)).

```

1565     <xsl:if test="count(db:chapter)=0">
1566       <xsl:text>% \label{</xsl:text>
1567       <xsl:value-of select="@xml:id"/>
1568       <xsl:text>}&#xa;</xsl:text>
1569       <xsl:call-template name="start-tag">
1570         <xsl:with-param name="identity">
1571           <xsl:value-of select="$doctype"/>
1572         </xsl:with-param>
1573       </xsl:call-template>
1574     </xsl:if>
1575     <xsl:text>% \clearpage\subsection*{Code for the </xsl:text>
1576     <xsl:apply-templates select="db:title/node()"/>

```

⁵The fixltx2e package is no longer needed as it is now coded into the kernel.

```

1577         <xsl:text>&#xa;</xsl:text>
1578     <xsl:apply-templates/>
1579     <xsl:if test="count(db:chapter) +
1580                 count(db:appendix) = 0">
1581         <xsl:call-template name="end-tag">
1582             <xsl:with-param name="identity">
1583                 <xsl:value-of select="$doctype"/>
1584             </xsl:with-param>
1585         </xsl:call-template>
1586     </xsl:if>

```

Again, if there were no chapters or appendixes, output the end-tag now.

db:part **Licence**

Once the code has been done, and before any standalone ancillary files are processed, we output the Licence as an appendix. As we are in a `<part>`, the `@role` attribute may contain the white-space-separated name[s] of [an] externally-generated *DocBook* document[s] to be included as an annexes (with their own generated markup at the `<chapter>` level to ensure formatting as appendices.

```

1587     <xsl:if test="not(//db:appendix)">
1588         <xsl:text>% \appendix&#xa;</xsl:text>
1589     </xsl:if>
1590     <!-- WITHDRAWN 2024-02-04
1591     <xsl:text>% \newgeometry{left=3cm}
1592 % \makeatletter\ltx@ifpackageloaded{fancyhdr}
1593 % {\lhead{\leavevmode\kern-\CPKrevmarg\footnotesize\rightmark}
1594 % \lfoot{\leavevmode\kern-\CPKrevmarg\footnotesize\leftmark}}
1595 % {\relax}\makeatother&#xa;</xsl:text>
1596     -->
1597     <xsl:apply-templates select="$licence"/>
1598     <xsl:if test="@role">
1599         <xsl:for-each
1600             select="tokenize(normalize-space(@role),
1601                             ' ')">
1602             <xsl:apply-templates
1603                 select="document(concat($appdir,'/',.,'.xml'))
1604                             /db:chapter"/>
1605         </xsl:for-each>
1606     </xsl:if>
1607 </xsl:when>

```

OMITTED: any delayed-output extractable files from the code part (these need no accompanying text as they're just re-given here for extraction).

db:part **Files**

Here we output any plain files that do not require DocTeX-style pre- or postamble.

There can in fact (`<xsl:otherwise>`) be a fourth kind of part, currently used only in the `uccthesi` package: a "data" part, currently undocumented.

```

1608     <xsl:when test="@xml:id='files'">
1609         <xsl:apply-templates mode="files"/>
1610     </xsl:when>

```

```

1611     <xsl:otherwise>
1612         <xsl:message>
1613             <xsl:text>WARNING: part element being used </xsl:text>
1614             <xsl:text>that is not doc/code/files</xsl:text>
1615         </xsl:message>
1616     </xsl:otherwise>
1617 </xsl:choose>
1618 <xsl:if test="count(following-sibling::db:part
1619     [@xml:id!='data'])=0">
1620     <xsl:text>% \Finale&#xa; </xsl:text>
1621 </xsl:if>
1622 </xsl:template>
1623
1624 <xsl:template match="db:part[@xml:id='code']/db:subtitle"/>

```

If there are only non-“data” parts following, we’re done.

db:chapter If this is the first chapter of a “code” or “files” part, output the start-tag. Check for the location of *AutoPackage* output.

```

1625 <xsl:template match="db:chapter[not(@condition='draft')]">
1626     <!-- non-empty only -->
1627     <xsl:if test="normalize-space(.)!=''">
1628         <xsl:choose>
1629             <!-- first one in code is THE package or class
1630                  so use that value as the tag -->
1631             <xsl:when
1632                 test="parent::db:part[@xml:id='code'] and
1633                     count(preceding-sibling::db:chapter)=0">
1634                 <xsl:call-template name="start-tag">
1635                     <xsl:with-param name="identity">
1636                         <xsl:value-of select="$doctype"/>
1637                     </xsl:with-param>
1638                 </xsl:call-template>
1639             </xsl:when>
1640             <!-- in files, use the ID as the tag -->
1641             <xsl:when
1642                 test="parent::db:part[@xml:id='files']
1643                     and
1644                     db:programlisting
1645                     and
1646                     @xml:id
1647                     and
1648                     @xlink:href">
1649                 <xsl:call-template name="start-tag"/>
1650             </xsl:when>
1651         </xsl:choose>
1652         <xsl:call-template name="checkpackages">
1653             <xsl:with-param name="pos" select="'before'"/>
1654             <xsl:with-param name="loc" select="."/>
1655         </xsl:call-template>

```

db:chapter Chapter starts

Chapters are the normal subdivision within each part, output as \LaTeX \sections; there may be an optional subtitle, version, and label.

```

1656      <xsl:if test="count(preceding-sibling::db:chapter)>0">
1657          <xsl:text>% \clearpage&#xa;</xsl:text>
1658      </xsl:if>
1659      <xsl:text>% \section</xsl:text>
1660      <!-- subtitle is used for short title -->
1661      <xsl:if test="db:subtitle">
1662          <xsl:text>[</xsl:text>
1663          <xsl:apply-templates select="db:subtitle/node()"/>
1664          <xsl:text>]</xsl:text>
1665      </xsl:if>
1666      <xsl:text>{</xsl:text>
1667      <xsl:apply-templates select="db:title/node()"/>
1668      <xsl:if test="db:subtitle">
1669          <xsl:text>~--- </xsl:text>
1670          <xsl:apply-templates select="db:subtitle/node()"/>
1671      </xsl:if>
1672      <xsl:if test="@version">
1673          <xsl:text> (v\thinspace{</xsl:text>
1674          <xsl:value-of select="@version"/>
1675          <xsl:text>)</xsl:text>
1676      </xsl:if>
1677      <xsl:text>}</xsl:text>
1678      <xsl:if test="@xml:id">
1679          <xsl:text>\label{</xsl:text>
1680          <xsl:value-of select="@xml:id"/>
1681          <xsl:text>}</xsl:text>
1682      </xsl:if>
1683      <xsl:text>&#xa;</xsl:text>
1684      <!-- echo the title/subtitle into comments
1685           IFF this is NOT a DTD, XML, or XSL file -->
1686      <xsl:if
1687          test="parent::db:part[@xml:id='code']
1688              and @xlink:href
1689              and @xml:id
1690              and not(substring-after(@xlink:href,'.')='dtd'
1691                  or substring-after(@xlink:href,'.')='xml'
1692                  or substring-after(@xlink:href,'.')='xsl')">
1693          <xsl:text>% \iffalse&#xa;</xsl:text>
1694          <xsl:value-of select="$sentinel"/>
1695          <xsl:text>&#xa;</xsl:text>
1696          <xsl:apply-templates select="db:title|db:subtitle"
1697              mode="readme"/>
1698          <xsl:text>% \fi&#xa;</xsl:text>
1699      </xsl:if>
1700      <!-- start the chapter -->
1701      <xsl:call-template name="makechapapp"/>
1702  </xsl:if>

```

The title is also output as a plaintext comment if this is a chapter in code with an

ID and a link for output that is not XML, XSLT, or DTD.

`makechapapp` The actual output of chapter content is handled by the `makechapapp` template (see [section 11.5.2 on page 249](#)).

`db:chapter` **Chapter ends**

Regardless of whether or not it had any content, close the DocTeX tag if this is the last chapter, or close the files tag if this chapter is in the “files” part.

```

1703     <xsl:choose>
1704         <xsl:when
1705             test="parent::db:part[@xml:id='code'] and
1706                 count(following-sibling::db:chapter)=0">
1707             <xsl:call-template name="end-tag">
1708                 <xsl:with-param name="identity">
1709                     <xsl:value-of select="$doctype"/>
1710                 </xsl:with-param>
1711             </xsl:call-template>
1712         </xsl:when>
1713         <xsl:when
1714             test="parent::db:part[@xml:id='files']
1715                 and
1716                 db:programlisting
1717                 and
1718                 @xml:id
1719                 and
1720                 @xlink:href">
1721             <xsl:call-template name="end-tag"/>
1722         </xsl:when>
1723     </xsl:choose>
1724 </xsl:template>

```

`db:appendix` Appendixes can be used in documentation, but are also used to hold annotated extractable ancillary files, as part of a “code” part.

If this is the first appendix, issue the `\appendix` LaTeX switch for the documentation, and adjust the automated cross-reference name for sections to ‘Appendix’. If this is in the code part, and the appendix also contains code, and has an ID and a link, tag it for extraction. Create the title, with optional subtitle and label.

```

1725 <!-- ANNOTATED FILES ARE APPENDIXES
1726      in the code part
1727      (in the file part, unannotated files are chapters)
1728      but you might have documentary appendices in the
1729      documentation -->
1730 <xsl:template match="db:appendix">
1731     <!-- ignore empty (placeholder) appendixes -->
1732     <xsl:if test="normalize-space(.)!=''">
1733         <!-- LaTeX switch to appendix handling in docs
1734             for the first one -->
1735         <xsl:if
1736             test="count(preceding-sibling::db:appendix)=0 and
1737                 ancestor::db:part/@xml:id='doc'">
1738             <xsl:text>% \appendix\labelformat{section}</xsl:text>

```

```

1739         <xsl:text>{Appendix~#1}&#xa;</xsl:text>
1740     </xsl:if>
1741     <!-- insert latexdoc start-tag for annotated code -->
1742     <xsl:if test="parent::db:part[@xml:id='code']
1743         and
1744         descendant::db:programlisting
1745         and
1746         @xlink:href
1747         and
1748         @xml:id">
1749         <xsl:call-template name="start-tag"/>
1750     </xsl:if>
1751     <!-- TITLE and SUBTITLE -->
1752     <xsl:text>% \clearpage&#xa;% \section</xsl:text>
1753     <xsl:if test="db:subtitle">
1754         <xsl:text>[</xsl:text>
1755         <xsl:apply-templates select="db:title/node()"/>
1756         <xsl:text>]</xsl:text>
1757     </xsl:if>
1758     <xsl:text>{</xsl:text>
1759     <xsl:apply-templates select="db:title/node()"/>
1760     <xsl:if test="db:subtitle">
1761         <xsl:text>~--- </xsl:text>
1762         <xsl:apply-templates select="db:subtitle/node()"/>
1763     </xsl:if>
1764     <xsl:text>}</xsl:text>
1765     <!-- LABEL -->
1766     <xsl:if test="@xml:id">
1767         <xsl:text>\label{</xsl:text>
1768         <xsl:value-of select="@xml:id"/>
1769         <xsl:text>}</xsl:text>
1770     </xsl:if>
1771     <xsl:text>&#xa;</xsl:text>

```

db:appendix Processing

makechapapp Finally, call makechapapp for the content as with chapters.

```

1772     <!-- start the appendix -->
1773     <xsl:call-template name="makechapapp"/>
1774     <xsl:if
1775         test="parent::db:part[@xml:id='code']
1776         and descendant::db:programlisting
1777         and @xlink:href and @xml:id">
1778         <xsl:call-template name="end-tag"/>
1779     </xsl:if>
1780 </xsl:if>
1781 </xsl:template>

```

startappcomment Prefix extracted files with a comment to show provenance, date, and time.

```

1782     <xsl:template name="startappcomment">
1783         <xsl:param name="scriptflag"/>
1784         <!-- non-script = 'no'
1785             script, first line not done yet = 'wait'

```

```

1786         script, first line done = "go" -->
1787     <!-- prefix extracted files with a comment to show
1788         provenance, date, and time (context is <appendix>);
1789         THE EXCEPTION is shell scripts where the comment is
1790         delayed until after the first line for hashbang
1791         reasons -->
1792     <xsl:if test="$scriptflag='no' or $scriptflag='go'">
1793         <xsl:text>% \begin{CPKcomment}{</xsl:text>
1794         <xsl:value-of select="ancestor-or-self::db:appendix/@xml:id"/>
1795         <xsl:text>-comment}</xsl:text>
1796         <xsl:text>\label{comment-</xsl:text>
1797         <xsl:value-of select="ancestor-or-self::db:appendix/@xml:id"/>
1798         <xsl:text>}&#xa;</xsl:text>
1799         <!-- IDENTITY COMMENT -->
1800         <xsl:text>% Identify the file origin,
1801 % date, time, etc.\par&#xa;</xsl:text>
1802         <xsl:text>% \begin</xsl:text>
1803         <xsl:text>{macrocode}&#xa;</xsl:text>
1804         <xsl:value-of select="sil:sentinel(.)"/>
1805         <xsl:text> </xsl:text>
1806         <xsl:value-of select="normalize-space(
1807         ancestor-or-self::db:appendix/db:title)"/>
1808         <xsl:if test="ancestor-or-self::db:appendix/db:subtitle">
1809             <xsl:text>: </xsl:text>
1810             <xsl:value-of select="normalize-space(
1811             ancestor-or-self::db:appendix/db:subtitle)"/>
1812         </xsl:if>
1813         <xsl:text> (from the </xsl:text>
1814         <xsl:value-of select="$docname"/>
1815         <xsl:text> </xsl:text>
1816         <xsl:value-of select="$doctype"/>
1817         <xsl:text> v.</xsl:text>
1818         <xsl:value-of select="$version"/>
1819         <xsl:text>.</xsl:text>
1820         <xsl:value-of select="$revision"/>
1821         <xsl:text>, </xsl:text>
1822         <xsl:value-of select="$date"/>
1823         <xsl:text>) </xsl:text>
1824         <xsl:value-of select="sil:termcommentchar(.)"/>
1825         <xsl:text>&#xa;</xsl:text>
1826         <xsl:value-of select="sil:sentinel(.)"/>
1827         <xsl:text> Generated by ClassPack v.</xsl:text>
1828         <xsl:value-of select="$thisversion"/>
1829         <xsl:text> (</xsl:text>
1830         <xsl:value-of select="$thisverdate"/>
1831         <xsl:text>) on </xsl:text>
1832         <!--
1833         <xsl:value-of
1834             select="format-dateTime(current-dateTime(),
1835             '[D] [MNn] [Y] at [H]:[m]:[s] ')/>
1836         -->
1837         <xsl:value-of
1838             select="format-dateTime(current-dateTime(),
1839             '[Y03]-[M01]-[D01]T[H01]:[m01]:[s01] ')/>

```

```

1840     <xsl:value-of select="sil:termcommentchar(.)"/>
1841     <xsl:text>&#xa;</xsl:text>
1842     <xsl:text>% \end</xsl:text>
1843     <xsl:text>{macrocode}&#xa;</xsl:text>
1844     <xsl:text>% These lines are automatically prepended to the file
1845 % and may be removed if necessary,
1846 % using a suitable editor.&#xa;</xsl:text>
1847     <xsl:text>% \end{CPKcomment}&#xa;</xsl:text>
1848   </xsl:if>
1849 </xsl:template>

```

This also gets called from the routine handling extracted fragments of a shell script, to preserve the hashbang line.

db:sect1 In sections, check the package insertions, and allow a new page. In code parts, add the title in comments as for chapters. After handling content, check package insertions again.

```

1850 <xsl:template match="db:sect1">
1851   <xsl:if test="normalize-space(.)!=''">
1852     <xsl:call-template name="checkpackages">
1853       <xsl:with-param name="pos" select="''before''"/>
1854       <xsl:with-param name="loc" select="''"/>
1855     </xsl:call-template>
1856     <xsl:if test="@condition='newpage'">
1857       <xsl:text>% \clearpage&#xa;</xsl:text>
1858     </xsl:if>
1859     <xsl:text>% \subsection</xsl:text>
1860     <!-- is @label for a * or an [alt]? -->
1861     <xsl:value-of select="@label"/>
1862     <xsl:text>{</xsl:text>
1863     <xsl:apply-templates select="db:title/node()"/>
1864     <xsl:text>}</xsl:text>
1865     <xsl:if test="@xml:id">
1866       <xsl:text>\label{</xsl:text>
1867       <xsl:value-of select="@xml:id"/>
1868       <xsl:text>}</xsl:text>
1869     </xsl:if>
1870     <xsl:text>&#xa;</xsl:text>
1871     <xsl:if test="ancestor::db:part[@xml:id='code']">
1872       <xsl:choose>
1873         <xsl:when test="ancestor-or-self::db:chapter">
1874           <xsl:text>% \iffalse&#xa;%&#xa;</xsl:text>
1875           <xsl:call-template name="makefence"/>
1876           <xsl:text>%&#xa;</xsl:text>
1877           <xsl:apply-templates mode="readme"
1878             select="db:title|db:subtitle"/>
1879           <xsl:if test="@xml:id='options'
1880             and not(db:annotation)
1881             and count(db:para)=1">
1882             <xsl:text>%&#xa;</xsl:text>
1883             <xsl:text>% There are no package options.&#xa;</xsl:text>
1884           </xsl:if>
1885           <xsl:text>% \fi&#xa;</xsl:text>

```

```

1886         </xsl:when>
1887         <xsl:otherwise>
1888             <!--
1889             <xsl:value-of select="sil:commentchar(.)"/>
1890             <xsl:apply-templates mode="readme"
1891             select="db:title|db:subtitle"/>
1892             <xsl:value-of select="sil:termcommentchar(.)"/>
1893             -->
1894         </xsl:otherwise>
1895     </xsl:choose>
1896 </xsl:if>
1897 <!-- process sect1 content -->
1898 <xsl:apply-templates/>
1899 <xsl:if test="@xml:id='options'">
1900     <xsl:text>% Now invoke the options.\par
1901 % \iffalse
1902 %% Now invoke the options.
1903 % \fi
1904 %     \begin{macrocode}
1905 \ExecuteOptions{</xsl:text>
1906     <xsl:value-of
1907     select="descendant::db:annotation
1908     [@condition='default']/@xreflabel"/>
1909     <xsl:text>}
1910     \ProcessOptions\relax&#xa;%     \end</xsl:text>
1911     <xsl:text>{macrocode}&#xa;</xsl:text>
1912 </xsl:if>
1913 <xsl:call-template name="checkpackages">
1914     <xsl:with-param name="pos" select="''after''"/>
1915     <xsl:with-param name="loc" select="."/>
1916 </xsl:call-template>
1917 </xsl:if>
1918 </xsl:template>

```

db:sect2 Works the same way as for <sect1>, but outputting content only for non-XML files (and XSL/DTD files).

```

1919 <xsl:template match="db:sect2">
1920     <xsl:if test="normalize-space(.)!=''">
1921         <xsl:call-template name="checkpackages">
1922             <xsl:with-param name="pos" select="''before''"/>
1923             <xsl:with-param name="loc" select="."/>
1924         </xsl:call-template>
1925         <xsl:text>% \subsubsection{</xsl:text>
1926         <xsl:apply-templates select="db:title/node()"/>
1927         <xsl:text>}</xsl:text>
1928         <xsl:if test="@xml:id">
1929             <xsl:text>\label{</xsl:text>
1930             <xsl:value-of select="@xml:id"/>
1931             <xsl:text>}</xsl:text>
1932         </xsl:if>
1933         <xsl:text>&#xa;</xsl:text>
1934         <xsl:if
1935             test="ancestor::db:part[@xml:id='code'] and

```

```

1936         ancestor::db:chapter/@xlink:href and
1937         ancestor::db:chapter/@xml:id and
1938         not(substring-after(
1939             ancestor::db:chapter/@xlink:href, '.')='dtd'
1940         or substring-after(
1941             ancestor::db:chapter/@xlink:href, '.')='xml'
1942         or substring-after(
1943             ancestor::db:chapter/@xlink:href, '.')='xsl'))">
1944     <xsl:text>% \iffalse&#xa;%% &#xa;</xsl:text>
1945     <xsl:apply-templates mode="readme"
1946         select="db:title|db:subtitle"/>
1947     <xsl:text>% \fi&#xa;</xsl:text>
1948 </xsl:if>
1949 <xsl:apply-templates/>
1950 <xsl:call-template name="checkpackages">
1951     <xsl:with-param name="pos" select="'after'"/>
1952     <xsl:with-param name="loc" select="."/>
1953 </xsl:call-template>
1954 </xsl:if>
1955 </xsl:template>

```

db:sect3 These output like <sect2> but as \paragraph commands, and add a colon when there is no existing terminating punctuation.

```

1956 <xsl:template match="db:sect3">
1957     <xsl:if test="normalize-space(.)!=''">
1958         <xsl:call-template name="checkpackages">
1959             <xsl:with-param name="pos" select="'before'"/>
1960             <xsl:with-param name="loc" select="."/>
1961         </xsl:call-template>
1962         <xsl:text>% \paragraph[</xsl:text>
1963         <xsl:apply-templates select="db:title/node()"/>
1964         <xsl:text>]{</xsl:text>
1965         <xsl:apply-templates select="db:title/node()"/>
1966         <xsl:if test="not(matches(normalize-space(db:title),
1967             '[:\-\[\].]{$'))">
1968             <xsl:text> :</xsl:text>
1969         </xsl:if>
1970         <xsl:text>}</xsl:text>
1971         <xsl:if test="@xml:id">
1972             <xsl:text>\label{</xsl:text>
1973             <xsl:value-of select="@xml:id"/>
1974             <xsl:text>}</xsl:text>
1975         </xsl:if>
1976         <xsl:text>&#xa;</xsl:text>
1977         <xsl:apply-templates/>
1978         <xsl:call-template name="checkpackages">
1979             <xsl:with-param name="pos" select="'after'"/>
1980             <xsl:with-param name="loc" select="."/>
1981         </xsl:call-template>
1982     </xsl:if>
1983 </xsl:template>

```

db:sect4 The \subparagraph does exactly the same, but with a different bullet.

```

1984 <xsl:template match="db:sect4">
1985   <xsl:if test="normalize-space(.)!=''">
1986     <xsl:call-template name="checkpackages">
1987       <xsl:with-param name="pos" select="'before'"/>
1988       <xsl:with-param name="loc" select="."/>
1989     </xsl:call-template>
1990     <xsl:text>% \subparagraph[</xsl:text>
1991     <xsl:apply-templates select="db:title/node()"/>
1992     <xsl:text>]{</xsl:text>
1993     <xsl:apply-templates select="db:title/node()"/>
1994     <xsl:if test="not(matches(normalize-space(db:title),
1995       '[:\-\[\].]$'))">
1996       <xsl:text> °</xsl:text>
1997     </xsl:if>
1998     <xsl:text>}</xsl:text>
1999     <xsl:if test="@xml:id">
2000       <xsl:text>\label{</xsl:text>
2001       <xsl:value-of select="@xml:id"/>
2002       <xsl:text>}</xsl:text>
2003     </xsl:if>
2004     <xsl:text>&#xa;</xsl:text>
2005     <xsl:apply-templates/>
2006     <xsl:call-template name="checkpackages">
2007       <xsl:with-param name="pos" select="'after'"/>
2008       <xsl:with-param name="loc" select="."/>
2009     </xsl:call-template>
2010   </xsl:if>
2011 </xsl:template>

```

db:section The <section> element can only be used in prelims (eg Prefaces).

```

2012 <xsl:template
2013   match="db:section/db:title
2014     [name(parent::db:section/parent::*[1])='']"/>
2015 <xsl:template match="db:section">
2016   <xsl:if test="@role='newpage'">
2017     <xsl:text>% \clearpage&#xa;</xsl:text>
2018   </xsl:if>
2019   <xsl:text>% \subsection*{</xsl:text>
2020   <xsl:apply-templates select="db:title/node()"/>
2021   <xsl:text>}</xsl:text>
2022   <xsl:if test="@xml:id">
2023     <xsl:text>\label{</xsl:text>
2024     <xsl:value-of select="@xml:id"/>
2025     <xsl:text>}</xsl:text>
2026   </xsl:if>
2027   <xsl:text>&#xa;</xsl:text>
2028   <xsl:apply-templates/>
2029 </xsl:template>

```

db:title [in [db:info/db:abstract/](#) and [db:preface/db:title](#) and
[db:preface/db:section/db:title](#) and [db:part/db:title](#) and

db:chapter/db:title and db:chapter/db:subtitle and
 db:chapter/db:sect1/db:title and db:chapter//db:sect2/db:title and
 db:chapter//db:sect3/db:title and db:chapter//db:sect4/db:title and
 db:appendix/db:title and db:appendix/db:sect1/db:title and
 db:appendix//db:sect2/db:title and db:appendix//db:sect3/db:title and
 db:appendix//db:sect4/db:title]

Titles within hierarchy elements are handled within the code for their parent element, so they must be excluded from being processed in document order.

```

2030 <xsl:template
2031     match="db:info/db:abstract/db:title |
2032           db:preface/db:title |
2033           db:preface/db:section/db:title |
2034           db:part/db:title |
2035           db:chapter/db:title | db:chapter/db:subtitle |
2036           db:chapter/db:sect1/db:title |
2037           db:chapter//db:sect2/db:title |
2038           db:chapter//db:sect3/db:title |
2039           db:chapter//db:sect4/db:title |
2040           db:appendix/db:title |
2041           db:appendix/db:subtitle |
2042           db:appendix/db:sect1/db:title |
2043           db:appendix//db:sect2/db:title |
2044           db:appendix//db:sect3/db:title |
2045           db:appendix//db:sect4/db:title"/>

```

11.4.3 Pool

db:acknowledgements Acknowledgements are just a subsection of the documentation. Like all sections/subsections, the location for handling the *AutoPackage* call is tested after processing to see if this is the place.

```

2046 <xsl:template match="db:acknowledgements">
2047   <xsl:text>% \subsection*{Acknowledgments}</xsl:text>
2048   <xsl:if test="@xml:id">
2049     <xsl:text>\label{</xsl:text>
2050     <xsl:value-of select="@xml:id"/>
2051     <xsl:text>}</xsl:text>
2052   </xsl:if>
2053   <xsl:text>&#xa;</xsl:text>
2054   <xsl:apply-templates/>
2055   <xsl:call-template name="checkpackages">
2056     <xsl:with-param name="pos" select="''after''"/>
2057     <xsl:with-param name="loc" select="''./''"/>
2058   </xsl:call-template>
2059 </xsl:template>

```

db:para This handles all paragraphs at any level. See below for details of automated list-item punctuation.

Normally (in the `<xsl:otherwise>`) a percent-space is output, and also if this paragraph is coming from a procedure from the `prepost.xml` file; but omit it

for the first paragraph in a list item or a table cell because it has already been handled by the container.

```

2060 <xsl:template match="db:para">
2061   <xsl:choose>
2062     <!-- paras from prepost.xml need shielding -->
2063     <xsl:when
2064       test="ancestor::db:procedure
2065         [@xml:id='prepackage'
2066         or
2067         @xml:id='postpackage']">
2068       <xsl:text>% </xsl:text>
2069     </xsl:when>
2070     <!-- first paras in list items do nothing WHY? -->
2071     <xsl:when
2072       test="(parent::db:listitem or parent::db:step) and
2073         count(preceding-sibling::db:para)=0">
2074       <xsl:text></xsl:text>
2075     </xsl:when>
2076     <!-- first paras in table cells likewise -->
2077     <xsl:when test="parent::db:entry and position()='1">
2078       <xsl:text></xsl:text>
2079     </xsl:when>
2080     <!-- all others get a shield -->
2081     <xsl:otherwise>
2082       <xsl:text>% </xsl:text>
2083     </xsl:otherwise>
2084   </xsl:choose>
2085   <!-- labels on paras (rare) get output at start -->
2086   <xsl:if test="@xml:id">
2087     <xsl:text>\label{</xsl:text>
2088     <xsl:value-of select="@xml:id"/>
2089     <xsl:text>}</xsl:text>
2090   </xsl:if>
2091   <!-- any typographic mods start here, end afterwards -->
2092   <xsl:if test="@remap">
2093     <xsl:text>{</xsl:text>
2094     <xsl:value-of select="@remap"/>
2095   </xsl:if>
2096   <!-- now the content -->
2097   <xsl:apply-templates/>

```

Add a label if there was an `@xml:id`, and add any manual reformatting in the `@remap` attribute, then process the content.

db:para **Auto-punctuation for list items**

Add punctuation if this is the sole or last paragraph in a list item, and it does *not* end with its own punctuation, and it is not followed by punctuation *immediately* at the end of its scope (eg manually-supplied punctuation after the end-tag of an inline list [within a paragraph]).

The punctuation added is a semicolon (to all but the last paragraph), or a full point (to the last paragraph).

None of this applies if the list is *inside* a paragraph (an inline list) because the inline features of L^AT_EX's `enumitem` package already handle it correctly.

```

2098     <!-- then the fun starts -->
2099     <xsl:choose>
2100         <!-- AUTOPUNCT: para elements in lists (NOT inline) which
2101             * do NOT end with their own punctuation
2102             AND
2103             * the container does NOT start with punctuation
2104             OR
2105             * there is nothing in scope after the container
2106         -->
2107     <xsl:when
2108         test="(parent::db:listitem[not(ancestor::db:para)]
2109             or
2110             parent::db:step[not(ancestor::db:para)])
2111         and
2112         not(contains('.,:;!?',
2113             substring(normalize-space(.),
2114             string-length(normalize-space(.))))))
2115         and
2116         (
2117             not(contains('.,:;!?',
2118                 substring(normalize-space(
2119                     (
2120                         ancestor::db:itemizedlist[1] |
2121                         ancestor::db:orderedlist[1] |
2122                         ancestor::db:variablelist[1] |
2123                         ancestor::db:procedure[1]
2124                     )[1]/
2125                     following-sibling::node()[1]),1,1)))
2126             or
2127             not( (
2128                 ancestor::db:itemizedlist[1] |
2129                 ancestor::db:orderedlist[1] |
2130                 ancestor::db:variablelist[1] |
2131                 ancestor::db:procedure[1]
2132                 )[1]/
2133                 following-sibling::node())
2134             )"
2135     <!-- if that is TRUE then munge the punct -->
2136     <xsl:choose>
2137         <!-- if last in the item, add semicolon or period -->
2138     <xsl:when
2139         test="not(following-sibling::db:para)"
2140     <xsl:choose>
2141     <xsl:when
2142         test="parent::db:listitem/
2143             following-sibling::db:listitem
2144             or
2145             ancestor::db:varlistentry[1]/
2146             following-sibling::db:varlistentry
2147             or
2148             parent::db:step/

```

```

2149         following-sibling::db:step">
2150         <xsl:text>;</xsl:text>
2151     </xsl:when>
2152     <xsl:otherwise>
2153         <xsl:text>.</xsl:text>
2154     </xsl:otherwise>
2155 </xsl:choose>
2156 </xsl:when>
2157 <!-- if not the last one, use a \par -->
2158 <xsl:otherwise>
2159     <xsl:text>\par </xsl:text>
2160 </xsl:otherwise>
2161 </xsl:choose>
2162 </xsl:when>
2163 <!-- paras in prepost.xml don't get autopunct -->
2164 <xsl:when
2165     test="(parent::db:step/parent::db:procedure
2166         [@xml:id='prepackage']
2167         or
2168         parent::db:step/parent::db:procedure
2169         [@xml:id='postpackage'])
2170         and
2171         not(following-sibling::db:para)">
2172     <xsl:text></xsl:text>
2173 </xsl:when>
2174 <!-- everything else gets a \par -->
2175 <xsl:otherwise>
2176     <xsl:text>\par</xsl:text>
2177 </xsl:otherwise>
2178 </xsl:choose>
2179 <!-- close off any formatting changes -->
2180 <xsl:if test="@remap">
2181     <xsl:text>}</xsl:text>
2182 </xsl:if>
2183 <xsl:if test="not(parent::db:note
2184     or
2185     parent::db:footnote
2186     or
2187     parent::db:entry)">
2188     <xsl:text>&#xa;</xsl:text>
2189     <xsl:if test="parent::db:entry">
2190         <xsl:text> </xsl:text>
2191     </xsl:if>
2192 </xsl:if>

```

Omit the `\par` if this is the last or sole paragraph in an auto-included documentation chunk from `prepost.xml`, otherwise output it. Terminate any local formatting.

In the case of notes, footnotes, and cell content we don't even want a newline, but we do need a space after `\par` in the case of cell content.

db:para Inline documentation in code

The inline documentation echo (except for footnotes, which get done inline in db2md.xsl) which gets added to the dtx file as comments, is done in readme mode.

```

2193   <!-- add plaintext documentation that will go through to the
2194         class or style file being generated. Don't do this here
2195         for the para of a warning or sidebar because that will
2196         still be inside a minipage. See the warning|sidebar
2197         template.
2198         IFF
2199         * it's in the code
2200         * it's not a footnote
2201         * it's in a chapter being output
2202         * it's in an ID'd chapter
2203         * it's not DTD/XML/XSL
2204         * it's not in a warning or sidebar
2205         (these are ANDed, so it means all of them!) -->
2206   <xsl:if
2207     test="ancestor::db:part[@xml:id='code']
2208           and not(parent::db:footnote)
2209           and ancestor::db:chapter[@xlink:href]
2210           and ancestor::db:chapter[@xml:id]
2211           and not(
2212             substring-after(ancestor::db:chapter
2213               /@xlink:href, '.')='dtd'
2214             or substring-after(ancestor::db:chapter
2215               /@xlink:href, '.')='xml'
2216             or substring-after(ancestor::db:chapter
2217               /@xlink:href, '.')='xsl')
2218           and not(parent::db:warning)
2219           and not(parent::db:sidebar)">
2220     <xsl:text>% \iffalse&#xa;</xsl:text>
2221     <xsl:apply-templates select="." mode="readme"/>
2222     <xsl:text>% \fi&#xa;</xsl:text>
2223   </xsl:if>
2224 </xsl:template>

```

Note that this only affects paragraphs in the “code” part which are *not* in footnotes but *are* in a chapter with a @href and @xml:id and are *not* part of a DTD, XML, or XSL document.

db:literal Literals are formatted in a monospace font except those flagged as \LaTeX , which are passed through as-is for processing.

```

2225   <xsl:template match="db:literal[not(@language='LaTeXe')
2226         and not(@language='LaTeX')]">
2227     <xsl:call-template name="avoidverb"/>
2228   </xsl:template>
2229
2230   <xsl:template match="db:literal[@language='LaTeXe'
2231         or @language='LaTeX']">
2232     <xsl:value-of select="."/>

```

2233 </xsl:template>

db:blockquote Block quotes use the quotation environment, with an optional citation and link at the end.

```

2234   <xsl:template match="db:blockquote">
2235     <xsl:text>% \begin{quotation}</xsl:text>
2236     <xsl:if test="@xml:id">
2237       <xsl:text>\label{</xsl:text>
2238       <xsl:value-of select="@xml:id"/>
2239       <xsl:text>}</xsl:text>
2240     </xsl:if>
2241     <xsl:text>\small\sffamily\parindent0pt
2242 % \parskip.5\baselineskip
2243 % \color{DarkBlue}</xsl:text>
2244     <xsl:if test="@remap">
2245       <xsl:value-of select="@remap"/>
2246     </xsl:if>
2247     <xsl:text>\noindent&#xa;</xsl:text>
2248     <xsl:apply-templates/>
2249     <xsl:if test="@linkend">
2250       <xsl:text>% \hfill\begin{group}</xsl:text>
2251       <xsl:call-template name="makeref"/>
2252       <xsl:text>\parfillskip=0pt\parskip=0pt\par</xsl:text>
2253       <xsl:text>\endgroup&#xa;</xsl:text>
2254     </xsl:if>
2255     <xsl:if test="@xlink:href">
2256       <xsl:text>% \hfill\begin{group
2257 % \scriptsize\color{Black}\url{</xsl:text>
2258       <xsl:value-of select="@xlink:href"/>
2259       <xsl:text>}\parfillskip=0pt\par</xsl:text>
2260       <xsl:text>\endgroup&#xa;</xsl:text>
2261     </xsl:if>
2262     <xsl:text>% \end{quotation}&#xa;</xsl:text>
2263   </xsl:template>

```

db:warning [db:warning and db:sidebar and db:warning/db:title and
db:sidebar/db:title and db:sidebar/db:address and db:bridgehead]

Warnings and sidebars are handled similarly, the difference being that sidebars use a drop-shadow box. The width of the box can be controlled with the @wordsize attribute, holding a percentage of the text width (default 80%).⁶

```

2264   <xsl:template match="db:warning | db:sidebar">
2265     <xsl:call-template name="sidewarn"/>
2266   </xsl:template>
2267
2268   <!-- named template used to allow "files" chapters
2269         to share this code from within mode="files" -->
2270
2271   <xsl:template name="sidewarn">

```

⁶Note that this is a different usage to that in tables, where the attribute affects *font size*.

```

2272     <xsl:text>% \par\begin{group
2273 % \fboxsep1em\centering&#xa;% \</xsl:text>
2274     <xsl:choose>
2275         <xsl:when test="name()='sidebar'">
2276             <xsl:text>shadowbox</xsl:text>
2277         </xsl:when>
2278         <xsl:when test="name()='warning'">
2279             <xsl:text>fbox</xsl:text>
2280         </xsl:when>
2281     </xsl:choose>
2282     <xsl:text>{\begin{minipage}}</xsl:text>
2283     <xsl:choose>
2284         <xsl:when test="@wordsize">
2285             <xsl:value-of
2286                 select="number(substring-before(@wordsize,'%'))
2287                     div 100"/>
2288         </xsl:when>
2289         <xsl:otherwise>
2290             <xsl:text>0.8</xsl:text>
2291         </xsl:otherwise>
2292     </xsl:choose>
2293     <xsl:text>\columnwidth}\sffamily
2294 % \raggedright\parindent0pt
2295 % \parskip=.5\baselineskip</xsl:text>
2296     <xsl:if test="@xml:id">
2297         <xsl:text>\label{</xsl:text>
2298         <xsl:value-of select="@xml:id"/>
2299         <xsl:text>}}</xsl:text>
2300     </xsl:if>
2301     <xsl:text>&#xa;</xsl:text>
2302     <xsl:apply-templates/>
2303     <xsl:text>% \end{minipage}}\par\end{group}&#xa;</xsl:text>
2304     <!-- now is the time to output the content in a form
2305         that will pass through to the sty or cls file. -->
2306     <!-- not working yet
2307     <xsl:text>% \iffalse&#xa;%% !!! </xsl:text>
2308     <xsl:apply-templates select="db:para" mode="readme"/>
2309     <xsl:text>% \fi&#xa;</xsl:text>
2310 -->
2311 </xsl:template>
2312
2313 <xsl:template match="db:warning/db:title
2314 |
2315         db:sidebar/db:title">
2316     <xsl:text>% \subsubsection*{\sffamily </xsl:text>
2317     <xsl:apply-templates/>
2318     <xsl:text>}&#xa;</xsl:text>
2319 </xsl:template>
2320
2321 <xsl:template match="db:sidebar/db:address">
2322     <xsl:text>% \par\raggedleft </xsl:text>
2323     <xsl:apply-templates/>
2324     <xsl:text>\par&#xa;</xsl:text>
2325 </xsl:template>

```

```

2326
2327 <xsl:template match="db:bridgehead">
2328   <xsl:text>% \subsubsection*{</xsl:text>
2329   <xsl:if test="parent::db:annotation">
2330     <!--
2331     <xsl:text>\protect\CPKrunningecho{</xsl:text>
2332     <xsl:value-of
2333       select="parent::db:annotation/@xreflabel"/>
2334     <xsl:text>}</xsl:text>
2335     -->
2336     <xsl:text>\llap{\color{OliveDrab}\LabelFont{</xsl:text>
2337     <xsl:value-of
2338       select="parent::db:annotation/@xreflabel"/>
2339     <xsl:text>}\quad}</xsl:text>
2340   </xsl:if>
2341   <xsl:apply-templates/>
2342   <xsl:text>}</xsl:text>
2343   <xsl:if test="@xml:id">
2344     <xsl:text>\label{</xsl:text>
2345     <xsl:value-of select="@xml:id"/>
2346     <xsl:text>}\addcontentsline{toc}</xsl:text>
2347     <xsl:text>{subparagraph}{</xsl:text>
2348     <xsl:value-of select="normalize-space(.)"/>
2349     <xsl:text>}</xsl:text>
2350   </xsl:if>
2351   <xsl:text>&#xa;</xsl:text>
2352 </xsl:template>

```

Titles, addresses, and bridgeheads (for embedded headings) are subsection-level elements or paragraphs.

db:annotation `[db:annotation with [not(@reuse)] [@role and @role!='' and @xreflabel] [not(@annotations)]] [not(ancestor::db:info)]]`

The `<annotation>` element is used to document a code fragment, often a whole macro definition. The `@role` attribute holds the type of operation being annotated (eg macro, environment, template, etc) and the `@xreflabel` attribute holds the name of the operation.

But first, if this is a colour declaration, we need to fake up a pre-definition for the colour, because the dox package appears to want it, and goes looking for it ahead of time (`@arch` holds the colourspace and `@audience` holds the colour definition).

db:annotation **TODO: Remove conflict in colour definition specs/attribs**

```

2353 <!-- The annotation element used here normally: not for @reuse,
2354       with a @role set to a known value (not "unused"), and
2355       with a non-null @xreflabel and NO @annotations and not
2356       in the <info> header (Phew!) -->
2357 <xsl:template match="db:annotation
2358                   [not(@reuse)]
2359                   [@role and @role!='unused' and
2360                   @xreflabel and @xreflabel!='']

```

```

2361             [not(@annotations)]
2362             [not(ancestor::db:info)]">
2363     <xsl:variable name="role" select="@role"/>
2364     <xsl:if test="$role='color'">
2365       <xsl:if test="not(@arch) or not(@audience)">
2366         <xsl:message>
2367           <xsl:text>CPK: no colour code given for </xsl:text>
2368           <xsl:value-of select="@xreflabel"/>
2369         </xsl:message>
2370       </xsl:if>
2371       <xsl:if test="parent::db:listitem">
2372         <xsl:text>&#xa;</xsl:text>
2373       </xsl:if>
2374       <xsl:text>% \definecolor{</xsl:text>
2375       <xsl:value-of select="@xreflabel"/>
2376       <xsl:text>}{</xsl:text>
2377       <xsl:value-of select="@arch"/>
2378       <xsl:text>}{</xsl:text>
2379       <xsl:value-of select="@audience"/>
2380       <xsl:text>}&#xa;</xsl:text>
2381     </xsl:if>

```

db:annotation **Start the definition**

The CPK prefix on defined names is local to *ClassPack* and keeps the namespace out of the way of other packages. The object name is given without a backslash, so this is added for macros and switches, or where the entry in the `prepost.xml` file has a command fulfilling the constraints listed below.

```

2382     <xsl:if test="parent::db:listitem and not(@role='color')">
2383       <xsl:text>&#xa;</xsl:text>
2384     </xsl:if>
2385     <xsl:text>% \begin{CPK</xsl:text>
2386     <xsl:value-of select="$role"/>
2387     <xsl:text>}{</xsl:text>
2388     <xsl:if test="$role='macro' or
2389                 $role='switch' or
2390                 $prepost//db:command
2391                 [starts-with(.,'\doxitem')]
2392                 [contains(.,concat('{CPK',$role,'}'))]
2393                 [contains(.,'macrolike')]">
2394       <xsl:text>\</xsl:text>
2395     </xsl:if>
2396     <xsl:value-of select="@xreflabel"/>
2397     <xsl:text>}</xsl:text>

```

db:annotation **Label the definition**

If this annotation is not called upon for reuse anywhere else, create a label for it with an prefix of the current `@role` attribute and the current filename (minus the extension).

```

2398     <xsl:if test="not(//db:annotation[@reuse=current()/@xml:id])">
2399       <xsl:text>\label{</xsl:text>
2400       <xsl:choose>

```

```

2401     <xsl:when test="@xml:id">
2402         <xsl:value-of select="@xml:id"/>
2403     </xsl:when>
2404     <xsl:otherwise>
2405         <xsl:value-of select="@role"/>
2406         <xsl:text>-</xsl:text>
2407         <xsl:value-of
2408             select="substring-before(db:programlisting[1]
2409                                     /@xlink:href,',' )"/>
2410         <xsl:text>-</xsl:text>
2411         <xsl:choose>
2412             <xsl:when test="contains(@xreflabel,':')">
2413                 <xsl:value-of
2414                     select="substring-after(@xreflabel,':')"/>
2415             </xsl:when>
2416             <xsl:otherwise>
2417                 <xsl:value-of select="@xreflabel"/>
2418             </xsl:otherwise>
2419         </xsl:choose>
2420         <xsl:if
2421             test="count(//db:annotation[@xreflabel=current()
2422                                     /@xreflabel])>1">
2423             <xsl:value-of
2424                 select="count(preceding::db:annotation
2425                             [@xreflabel=current()/@xreflabel])+1"/>
2426         </xsl:if>
2427     </xsl:otherwise>
2428 </xsl:choose>
2429 <xsl:text>}</xsl:text>
2430 </xsl:if>
2431 <xsl:text>&#xa;</xsl:text>

```

db:annotation Add to Table of Contents

If the @arch attribute is used, use it to specify the level at which the ToC entry will be made (eg section, paragraph, etc ****CONFLICT****).

```

2432 <!--
2433     <xsl:if test="@arch">
2434         <xsl:text>% \addcontentsline{toc}{</xsl:text>
2435         <xsl:value-of select="@arch"/>
2436         <xsl:text>}\texttt{</xsl:text>
2437     <xsl:choose>
2438         <xsl:when test="contains(@xreflabel,':')">
2439             <xsl:value-of
2440                 select="substring-after(@xreflabel,':')"/>
2441         </xsl:when>
2442         <xsl:otherwise>
2443             <xsl:value-of select="@xreflabel"/>
2444         </xsl:otherwise>
2445     </xsl:choose>
2446     <xsl:text>}</xsl:text>
2447     <xsl:if test="contains(@condition,'|')">
2448         <xsl:text> and others</xsl:text>
2449     </xsl:if>

```

```

2450      <xsl:text>}&#xa;</xsl:text>
2451    </xsl:if>
2452  -->

```

db:annotation Additional path conditions

If the `<condition>` is used, it should hold the full `@match` value of the XSLT template being matched, including alternation, predicates and namespaces.

```

2453    <xsl:if test="@condition">
2454      <xsl:text>% \begingroup\raggedright[</xsl:text>
2455      <xsl:for-each
2456        select="tokenize(replace(normalize-space(@condition),
2457          'xxx:','),'\|')">
2458        <xsl:variable name="subject">
2459          <xsl:choose>
2460            <xsl:when test="contains(.,'[')">
2461              <xsl:value-of
2462                select="replace(substring-before(
2463                  normalize-space(.),'['),'/','/'>"/>
2464            </xsl:when>
2465            <xsl:otherwise>
2466              <xsl:value-of select="normalize-space(.)"/>
2467            </xsl:otherwise>
2468          </xsl:choose>
2469        </xsl:variable>
2470        <xsl:variable name="predicate">
2471          <xsl:if test="contains(.,'[')">
2472            <xsl:value-of
2473              select="concat('[',
2474                substring-after(normalize-space(.),'['))"/>
2475          </xsl:if>
2476        </xsl:variable>
2477        <xsl:if test="position()>1">
2478          <xsl:text>and </xsl:text>
2479        </xsl:if>
2480        <xsl:choose>
2481          <xsl:when
2482            test="contains($subject,'/') and position()=1">
2483            <xsl:text>in \url{</xsl:text>
2484            <xsl:variable name="lastone"
2485              select="concat(tokenize($subject,'/')
2486                [position()=last()],'$')"/>
2487            <xsl:value-of
2488              select="replace($subject,$lastone,'')"/>
2489            <xsl:text>} </xsl:text>
2490          </xsl:when>
2491          <xsl:otherwise>
2492            <xsl:text>\url{</xsl:text>
2493            <xsl:value-of select="$subject"/>
2494            <xsl:text>} </xsl:text>
2495          </xsl:otherwise>
2496        </xsl:choose>
2497        <xsl:if test="$predicate!=''">
2498          <xsl:text>with \url{</xsl:text>

```

```

2499         <xsl:value-of select="$predicate"/>
2500         <xsl:text>} </xsl:text>
2501     </xsl:if>
2502     <xsl:text>&#xa;% </xsl:text>
2503 </xsl:for-each>
2504     <xsl:text>]\par\endgroup&#xa;</xsl:text>
2505 </xsl:if>

```

db:annotation **End the definition**

Apply content, and terminate the CPK block.

```

2506     <xsl:apply-templates/>
2507     <xsl:text>% \end{CPK</xsl:text>
2508     <xsl:value-of select="$role"/>
2509     <xsl:text>}&#xa;</xsl:text>
2510 </xsl:template>

```

db:constraintdef [db:constraintdef with [@reuse]]

Reuse in annotated Appendixes (extractable code) of previously-defined macros in <annotation> elements can be done using a <constraintdef> element. The @reuse is declared as IDREFS, so it allows the IDs of multiple <annotation> elements to be referenced (space-separated).

Find out which appendix we're in, and the type of document we're writing, and write a comment in the .dtx file about it.

Iterate over the packages specified, excluding any marked for omission for this application, adding in any documentation from the prepost.xml file, and any added locally in the @role attribute.

Then write the \RequirePackage in the same way as done for normally-included packages.

db:constraintdef **TODO: Make this a routine**

```

2511 <xsl:template match="db:constraintdef[@reuse]">
2512     <xsl:variable name="thisapp"
2513         select="ancestor::db:appendix/@xml:id"/>
2514     <xsl:variable name="thisdoctype"
2515         select="substring-after(
2516             ancestor::db:appendix/@xlink:href,','.'')"/>
2517     <xsl:text>% This is a subset of packages </xsl:text>
2518     <xsl:text>required for the \texttt{.}</xsl:text>
2519     <xsl:value-of select="$filetype"/>
2520     <xsl:text>} file.&#xa;</xsl:text>
2521     <xsl:for-each
2522         select="//db:constraintdef[@xml:id=current()/@reuse]
2523             //descendant::db:seglistitem
2524             [not(contains(@omit,$thisapp))]">
2525         <xsl:apply-templates
2526             select="$prepost/db:procedure[@xml:id='prepackage']
2527                 /db:step[contains(@condition,$thisdoctype)]

```

```

2528         [@remap=current()/db:seg]/db:para"/>
2529     <xsl:if test="@role">
2530         <xsl:text>% </xsl:text>
2531         <xsl:value-of select="normalize-space(@role)"/>
2532         <xsl:text>&#xa;</xsl:text>
2533     </xsl:if>
2534     <xsl:text>% </xsl:text>
2535     <xsl:text>\begin</xsl:text>
2536     <xsl:text>{macrocode}</xsl:text>
2537     <xsl:for-each select="db:seg">
2538         <!-- should only be one of these per seglistitem -->
2539         <xsl:text>&#xa;\RequirePackage</xsl:text>
2540         <xsl:if test="@role">
2541             <xsl:text>[</xsl:text>
2542             <xsl:value-of select="normalize-space(@role)"/>
2543             <xsl:text>]</xsl:text>
2544         </xsl:if>
2545         <xsl:text>{</xsl:text>
2546         <xsl:value-of select="normalize-space(.)"/>
2547         <xsl:text>}</xsl:text>
2548         <xsl:if test="@version">
2549             <xsl:text>[</xsl:text>
2550             <xsl:value-of select="translate(@version,'-','/')"/>
2551             <xsl:text>]</xsl:text>
2552         </xsl:if>
2553         <xsl:text>&#xa;% </xsl:text>
2554         <xsl:text>\end</xsl:text>
2555         <xsl:text>{macrocode}&#xa;</xsl:text>
2556     <xsl:apply-templates
2557         select="$prepost/db:procedure[@xml:id='postpackage']
2558             /db:step[contains(@condition,$thisdoctype)]
2559             [@remap=current()/db:seg]/db:para"/>
2560     </xsl:for-each>
2561 </xsl:for-each>
2562 </xsl:template>

```

db:annotation [**db:annotation** with [**@reuse**]

This template is for the case of reusability of existing annotated code, eg in a package when it is already defined for a class. The @reuse attribute is IDREFS, so it can hold a space-separated list of @xml:ids pointing at the <annotation>s to reuse.

```

2563     <xsl:template match="db:annotation[@reuse]">
2564         <xsl:for-each select="tokenize(normalize-space(@reuse),' ')">
2565             <xsl:variable name="block" select="."/>
2566             <xsl:apply-templates
2567                 select="$thisdoc//db:annotation[@xml:id=$block]"/>
2568         </xsl:for-each>
2569     </xsl:template>

```

db:remark The <remark> element holds comments inserted at change-points which need to

be preserved for accumulation by DocTeX's change-documentation mechanism.

```

2570 <xsl:template match="db:remark">
2571   <xsl:text>% \changes{v</xsl:text>
2572   <xsl:value-of select="@version"/>
2573   <xsl:text>}}{</xsl:text>
2574   <xsl:choose>
2575     <xsl:when test="@revision">
2576       <xsl:value-of select="translate(@revision,'-','/')"/>
2577     </xsl:when>
2578     <xsl:otherwise>
2579       <xsl:value-of
2580         select="translate(
2581           //db:revhistory/db:revision
2582             [@version=current()/@version]
2583             /db:date/@YYYY-MM-DD,'-','/')"/>
2584     </xsl:otherwise>
2585   </xsl:choose>
2586   <xsl:text>}}{</xsl:text>
2587   <xsl:value-of select="normalize-space(.)"/>
2588   <xsl:text>}&#xa;</xsl:text>
2589   <!-- also reproduce the comments in plaintext as comments -->
2590   <xsl:if
2591     test="ancestor::db:part[@xml:id='code']
2592       and ancestor::db:chapter/@xlink:href
2593       and ancestor::db:chapter/@xml:id
2594       and not(substring-after(
2595         ancestor::db:chapter/@xlink:href,'.')='dtd'
2596         or substring-after(
2597         ancestor::db:chapter/@xlink:href,'.')='xml'
2598         or substring-after(
2599         ancestor::db:chapter/@xlink:href,'.')='xsl')">
2600     <xsl:text>% \iffalse&#xa;</xsl:text>
2601     <xsl:value-of select="$sentinel"/>
2602     <xsl:text>CHANGE for v</xsl:text>
2603     <xsl:value-of select="@version"/>
2604     <xsl:text> (</xsl:text>
2605     <xsl:choose>
2606       <xsl:when test="@revision">
2607         <xsl:value-of select="translate(@revision,'-','/')"/>
2608       </xsl:when>
2609       <xsl:otherwise>
2610         <xsl:value-of
2611           select="translate(
2612             //db:revhistory/db:revision
2613               [@version=current()/@version]
2614               /db:date/@YYYY-MM-DD,'-','/')"/>
2615         </xsl:otherwise>
2616       </xsl:choose>
2617     <xsl:text>):</xsl:text>
2618     <xsl:text>&#xa;</xsl:text>
2619     <xsl:apply-templates select="." mode="readme"/>
2620     <xsl:text>% \fi&#xa;</xsl:text>
2621   </xsl:if>

```

2622 </xsl:template>

db:programlisting This template manages the reproduction of code listings, both from external files or from verbatim chunks in the document. The listings themselves are produced by the makelistings named template. There are a *lot* of controls, expressed as attributes:

- † @annotations for the list of keywords to be highlighted
- † @arch for signalling framed output
- @condition set when a listing is to be ignored (only used for documentation, not for the class or package)
- @conformance for how the listing is included (values "frag" or "file")
- † @endinglinenumber end boundary of a fragmentary listing
- @language any of the listings languages (as modified in *ClassPack*)
- † @linenumbering turns line numbering on
- † @remap font control for highlighting keywords
- @role specifies a title for the listing (rare)
- @startinglinenumber start boundary of a fragmentary listing, or name of a file fragment
- † @wordsize specifies listing font size
- @xlink:href name of an external file to list
- @xlink:show used in [section 11.6 on page 281](#) (and possibly in the code on [page 132](#)) to get the filename for installation.
- +xml:id use for a label

makelisting Those marked with a dagger (†) are only accessed by the named template makelisting (see the code on [page 230](#)).

The whole template is a choice between *a*) its principal use in generating DoCT_EX macrocode; *b*) listings in documentation *which contain embedded markup* (eg mnemonic tokens requiring italicisation); *c*) code examples not required in class or package output; and *d*) other uses in documentation or ancillary files

```

2623     <xsl:template match="db:programlisting">
2624       <!-- Handles code in annotated and unannotated blocks -->
2625       <xsl:variable name="curapp"
2626         select="generate-id(ancestor::db:appendix[1])"/>
2627       <!-- count the prior listing blocks within this appendix -->
2628       <xsl:variable name="nthproglis"
2629         select="count(preceding::db:programlisting
2630                       [generate-id(ancestor::db:appendix[1])=$curapp]) + 1"/>

```

db:programlisting **Generating macrocode fragments for the package or class**

This occurs only in the "code" part or in <annotation> elements with an @xreflabel attribute.

Any fragments required as external files specified in the "code" part (eg DTDs, XSLT, etc, get output here as extractables at the end of the .dtx file, as separate DocT_EX tagging can't be nested inside class or package tags. Otherwise (see end of section) they just get output as-is (trimmed fore and aft).

Due to a bug in the listings package when handling string-delimited fragments in very large files, it is necessary to split the file into fragments and process them individually (as is done for this document). The *chunk*(1) utility does the splitting.

```

2631 <!-- programlisting within an annotation
2632      OR anywhere in the code part -->
2633 <xsl:choose>
2634   <xsl:when test="(parent::db:annotation[@xreflabel]
2635                  or
2636                  ancestor::db:part[@xml:id='code'])
2637                  and not(@condition='ignore')">
2638     <xsl:if test="@xml:id">
2639       <xsl:text>% \label{</xsl:text>
2640       <xsl:value-of select="@xml:id"/>
2641       <xsl:text>}&#xa;</xsl:text>
2642     </xsl:if>
2643   <xsl:choose>
2644     <!-- EXTERNAL FRAGMENT whatever the language -->
2645     <xsl:when test="@xlink:href">
2646       <xsl:choose>
2647         <!-- frag means file has been pre-split into pieces -->
2648         <xsl:when test="@conformance='frag'">
2649           <!-- get the original source filename, strip dir -->
2650           <xsl:variable name="fn">
2651             <xsl:choose>
2652               <xsl:when test="contains(@xlink:href, '/')">
2653                 <xsl:value-of
2654                   select="tokenize(@xlink:href, '/')
2655                           [position()=last()]" />
2656               </xsl:when>
2657               <xsl:otherwise>
2658                 <xsl:value-of select="@xlink:href" />
2659               </xsl:otherwise>
2660             </xsl:choose>
2661           </xsl:variable>
2662           <!-- get the path to use on the fragment -->
2663           <xsl:variable name="dir">
2664             <xsl:if test="contains(@xlink:href, '/')">
2665               <xsl:for-each
2666                 select="tokenize(@xlink:href, '/')
2667                       [not(position()=last())]">
2668                 <xsl:value-of select="."/>
2669                 <xsl:text>/</xsl:text>
2670               </xsl:for-each>
2671             </xsl:if>
2672           </xsl:variable>
2673           <!-- construct the fragment filename, format is
2674            directory path, slash
2675            filename minus the extension
2676            MINUS
2677            fragname (@startinglinenumber string)

```

```

2678         DOT
2679         value of @conformance ('frag')
2680         -->
2681     <xsl:variable name="fragfn">
2682         <xsl:value-of select="$appdir"/>
2683         <xsl:text>/</xsl:text>
2684         <xsl:value-of select="$dir"/>
2685         <xsl:value-of
2686             select="substring-before($fn, '.')"/>
2687         <xsl:text>-</xsl:text>
2688         <xsl:value-of
2689             select="@startinglinenumber"/>
2690         <xsl:text>.</xsl:text>
2691         <xsl:value-of select="@conformance"/>
2692     </xsl:variable>
2693     <!-- pass to the routine that decide if this is a shell
2694          script or not, and if it's the first fragment or
2695          not, and outputs the hashbang first, THEN the
2696          comment and then the rest of the file.
2697          Or just the file. -->
2698     <xsl:call-template name="extinc">
2699         <xsl:with-param name="extfile" select="$fragfn"/>
2700         <xsl:with-param name="nthproglis"
2701             select="$nthproglis"/>
2702     </xsl:call-template>
2703 </xsl:when>
2704 <!-- non-frag PART OF FILE RETRIEVED FROM WHOLE FILE -->
2705 <xsl:when test="@startinglinenumber
2706             and
2707             @endinglinenumber">
2708     <xsl:call-template name="makelisting"/>
2709 </xsl:when>
2710 <!-- WHOLE FILE NOT A FRAGMENT -->
2711 <xsl:otherwise>
2712     <xsl:variable name="wholefn">
2713         <xsl:value-of select="$appdir"/>
2714         <xsl:text>/</xsl:text>
2715         <xsl:value-of select="@xlink:href"/>
2716     </xsl:variable>
2717     <!-- pass to the routine that decide if this is a shell
2718          script or not, and if it's the first fragment or
2719          not, and outputs the hashbang first, THEN the
2720          comment and then the rest of the file.
2721          Or just the file. -->
2722     <xsl:call-template name="extinc">
2723         <xsl:with-param name="extfile" select="$wholefn"/>
2724         <xsl:with-param name="nthproglis"
2725             select="$nthproglis"/>
2726     </xsl:call-template>
2727 </xsl:otherwise>
2728 </xsl:choose>
2729 </xsl:when>
2730 <!-- NORMAL EMBEDDED LISTING -->
2731 <xsl:otherwise>

```

```

2732         <xsl:text>%      </xsl:text>
2733         <xsl:text>\begin</xsl:text>
2734         <xsl:text>{macrocode}&#xa;</xsl:text>
2735         <xsl:call-template name="lrtrim">
2736           <xsl:with-param name="text" select="."/>
2737         </xsl:call-template>
2738         <xsl:text>&#xa;%      </xsl:text>
2739         <xsl:text>\end</xsl:text>
2740         <xsl:text>{macrocode}&#xa;</xsl:text>
2741       </xsl:otherwise>
2742     </xsl:choose>
2743   </xsl:when>

```

db:programlisting **Interpretable content in documentation (embedded markup)**

Where the code has embedded markup (@token or @xref) requiring syntactic representation, we use the Verbatim environment from the fancyverb package.

```

2744   <!-- programlisting containing token or xref (!)
2745         ILLUSTRATIONS ONLY, NOT EXTRACTABLE CODE -->
2746   <xsl:when test="db:token or db:xref">
2747     <xsl:text>% \iffalse
2748   %&lt;*ignore>
2749   % \fi
2750   % \begin{Verbatim}[frame=single,framesep=1em,
2751   %   commandchars=\\\{\},fontsize=</xsl:text>
2752     <xsl:choose>
2753       <xsl:when test="@wordsize and not(contains(@wordsize,'/'))">
2754         <xsl:value-of select="@wordsize"/>
2755       </xsl:when>
2756       <xsl:otherwise>
2757         <xsl:text>\small</xsl:text>
2758       </xsl:otherwise>
2759     </xsl:choose>
2760     <xsl:if test="@xml:id and @role">
2761       <xsl:text>,label=</xsl:text>
2762       <xsl:value-of select="@xml:id"/>
2763       <xsl:text>,title={</xsl:text>
2764       <xsl:value-of select="normalize-space(@role)"/>
2765       <xsl:text>}</xsl:text>
2766     </xsl:if>
2767     <xsl:text>]&#xa;</xsl:text>
2768     <xsl:apply-templates/>
2769     <xsl:text>&#xa;% \end{Verbatim}
2770   % \iffalse
2771   %&lt;/ignore>
2772   % \fi&#xa;</xsl:text>
2773   </xsl:when>

```

db:programlisting **Examples in code that are *not* for inclusion in the class or package**

This works the same way but using <xsl:value-of> to output the content as-is, instead of <xsl:apply-templates> to parse it.

```

2774   <!-- programlisting set to be ignored (!)

```

```

2775      ILLUSTRATIONS ONLY, NOT EXTRACTABLE CODE -->
2776      <xsl:when test="@condition='ignore'">
2777        <xsl:text>% \iffalse
2778      %&lt;*ignore>
2779      % \fi
2780      % \begin{group}\advance\hsize by-1em
2781      % \begin{Verbatim}[frame=single,framesep=1em,
2782      %   fontsize=</xsl:text>
2783        <xsl:choose>
2784          <xsl:when test="@wordsize and not(contains(@wordsize,'/'))">
2785            <xsl:value-of select="@wordsize"/>
2786          </xsl:when>
2787          <xsl:otherwise>
2788            <xsl:text>\small</xsl:text>
2789          </xsl:otherwise>
2790        </xsl:choose>
2791        <xsl:text>]&#xa;</xsl:text>
2792        <xsl:call-template name="ltrim">
2793          <xsl:with-param name="text" select="."/>
2794        </xsl:call-template>
2795        <xsl:text>
2796      % \end{Verbatim}
2797      % \endgroup
2798      % \iffalse
2799      %&lt;/ignore>
2800      % \fi&#xa;</xsl:text>
2801    </xsl:when>

```

db:programlisting **Use in documentation or ancillary files (default)**

In otherwise unadorned examples, eg non- \LaTeX , non- \DocTeX , this allows use of `<userinput>` subelements.

```

2802    <!-- ALL OTHER PROGRAMLISTINGS
2803          especially those using start\end from a whole file -->
2804    <xsl:otherwise>
2805      <xsl:call-template name="makelisting"/>
2806    </xsl:otherwise>
2807  </xsl:choose>
2808 </xsl:template>

```

extinc This named template handles the case of a file being input for extraction in fragments which is a shell script, and which therefore requires the first line (which contains the ‘hashbang’ directive to the operating system) to be preserved as the first line, and not prefixed by the comment lines usually imposed by *ClassPack*.

```

2809    <!-- EXTERNAL FILE INCLUSION -->
2810    <xsl:template name="extinc">
2811      <!-- context is a <programlisting> element in an <appendix> -->
2812      <xsl:param name="extfile"/>
2813      <xsl:param name="nthproglst"/>
2814      <xsl:choose>
2815        <!-- if it's a first fragment, AND it's a shell script,
2816              output the hashbang line, then the CP header

```

```

2817         followed by the rest -->
2818         <xsl:when test="ancestor::db:part[@xml:id='code']
2819             and
2820             ancestor::db:appendix[@arch='script'
2821                 and @userlevel='bash']
2822             and
2823             $nthproglis=1">
2824             <!-- output first line -->
2825             <xsl:text>% \begin</xsl:text>
2826             <xsl:text>{macrocode}&#xa;</xsl:text>
2827             <xsl:value-of
2828                 select="substring-before(unparsed-text($extfile),'&#xa;')"/>
2829             <xsl:text>&#xa;% \end</xsl:text>
2830             <xsl:text>{macrocode}&#xa;</xsl:text>
2831             <xsl:text>% \label{</xsl:text>
2832             <xsl:if test="number(@startinglinenumber)">
2833                 <xsl:text>frag-</xsl:text>
2834             </xsl:if>
2835             <xsl:value-of select="@startinglinenumber"/>
2836             <xsl:text>}&#xa;</xsl:text>
2837             <!-- go back and do the header comment -->
2838             <xsl:call-template name="startappcomment">
2839                 <xsl:with-param name="scriptflag">
2840                     <xsl:text>go</xsl:text>
2841                 </xsl:with-param>
2842             </xsl:call-template>
2843             <!-- then the remainder -->
2844             <xsl:text>% \begin</xsl:text>
2845             <xsl:text>{macrocode}&#xa;</xsl:text>
2846             <xsl:call-template name="lrtrim">
2847                 <xsl:with-param name="text"
2848                     select="substring-after(unparsed-text($extfile),'&#xa;')"/>
2849             </xsl:call-template>
2850             <xsl:text>&#xa;% \end</xsl:text>
2851             <xsl:text>{macrocode}&#xa;</xsl:text>
2852         </xsl:when>
2853         <!-- ELSE either it's not a bash script, or not a first file -->
2854         <xsl:otherwise>
2855             <!-- otherwise just output it -->
2856             <xsl:text>% \begin</xsl:text>
2857             <xsl:text>{macrocode}&#xa;</xsl:text>
2858             <xsl:call-template name="lrtrim">
2859                 <xsl:with-param name="text"
2860                     select="unparsed-text($extfile)"/>
2861             </xsl:call-template>
2862             <xsl:text>&#xa;% \end</xsl:text>
2863             <xsl:text>{macrocode}&#xa;</xsl:text>
2864         </xsl:otherwise>
2865     </xsl:choose>
2866 </xsl:template>

```

It does this by testing for the type of file, and that this is the first fragment of it. In this case it outputs the first line, calls the startappcomment named template, and then outputs the remainder of the fragment.

db:userinput User input example needing emphasis (*outside* the program listing).

```

2867 <xsl:template
2868     match="db:userinput[not(parent::db:programlisting)]">
2869     <xsl:text>\emph{</xsl:text>
2870     <xsl:apply-templates/>
2871     <xsl:text>}</xsl:text>
2872 </xsl:template>

```

11.4.4 Lists

db:itemizedlist Bulleted lists. If this is inside a paragraph (inline list), start with a newline. If there's a title, make it a paragraph* (subparagraph* if this is a nested list). For an inline list, use `enumerate*` with automated punctuation, otherwise just a normal `itemize`. For compact spacing, set the `noitemsep` option.

```

2873 <xsl:template match="db:itemizedlist">
2874     <xsl:if test="parent::db:para">
2875         <xsl:text>&#xa;</xsl:text>
2876     </xsl:if>
2877     <xsl:if test="db:title">
2878         <xsl:text>% \</xsl:text>
2879         <xsl:if test="ancestor::db:itemizedlist
2880                     or
2881                     ancestor::db:orderedlist">
2882             <xsl:text>sub</xsl:text>
2883         </xsl:if>
2884         <xsl:text>paragraph*</xsl:text>
2885         <xsl:apply-templates select="db:title/node()"/>
2886         <xsl:text>}&#xa;</xsl:text>
2887     </xsl:if>
2888     <xsl:choose>
2889         <xsl:when test="parent::db:para">
2890             <xsl:text>% \begin{enumerate}</xsl:text>
2891             <xsl:text>[label=\emph{\alph*}],</xsl:text>
2892             <xsl:text>itemjoin={};</xsl:text>
2893             <xsl:text>itemjoin*={}; or }</xsl:text>
2894         </xsl:when>
2895         <xsl:otherwise>
2896             <xsl:text>% \begin{itemize}</xsl:text>
2897             <xsl:if test="@spacing='compact'">
2898                 <xsl:text>[noitemsep]</xsl:text>
2899             </xsl:if>
2900         </xsl:otherwise>
2901     </xsl:choose>
2902     <xsl:if test="@xml:id">
2903         <xsl:text>\label{</xsl:text>
2904         <xsl:value-of select="@xml:id"/>
2905         <xsl:text>}</xsl:text>
2906     </xsl:if>
2907     <xsl:text>&#xa;</xsl:text>
2908     <xsl:apply-templates/>
2909     <xsl:choose>
2910         <xsl:when test="parent::db:para">

```

```

2911         <xsl:text>% \end{enumerate*}</xsl:text>
2912     </xsl:when>
2913     <xsl:otherwise>
2914         <xsl:text>% \end{itemize}</xsl:text>
2915     </xsl:otherwise>
2916 </xsl:choose>
2917 <xsl:if test="not(parent::db:para or parent::db:note)">
2918     <xsl:text>&#xa;</xsl:text>
2919 </xsl:if>
2920 </xsl:template>
2921
2922 <xsl:template match="db:itemizedlist/db:title"/>
2923
2924 <xsl:template
2925     match="db:varlistentry |
2926           db:varlistentry/db:listitem |
2927           db:authorgroup">
2928     <xsl:apply-templates/>
2929 </xsl:template>

```

Suppress the title when it is presented in document order, and pass through everything in variable list entries.

db:orderedlist [db:orderedlist and db:procedure and db:simplelist with
[*@arch='enumerate'*]]

Numbered lists (including Procedures and simple lists with the *@arch* attribute set to say so). For inline lists, set additional armour. Allow a two-column layout, and allow for suspend/resume. Otherwise similar to the `<itemizedlist>`.

```

2930 <xsl:template match="db:orderedlist |
2931                   db:procedure |
2932                   db:simplelist[@arch='enumerate']">
2933     <xsl:variable name="armour">
2934         <xsl:choose>
2935             <xsl:when test="parent::db:para">
2936                 <xsl:text></xsl:text>
2937             </xsl:when>
2938             <xsl:otherwise>
2939                 <xsl:text>%</xsl:text>
2940             </xsl:otherwise>
2941         </xsl:choose>
2942     </xsl:variable>
2943     <!-- check format for twocol wrapper -->
2944     <xsl:if test="@role='twocol'">
2945         <xsl:value-of select="$armour"/>
2946         <xsl:text> \begin{multicols}{2}&#xa;</xsl:text>
2947     </xsl:if>
2948     <xsl:if test="db:title">
2949         <xsl:text>% \</xsl:text>
2950     <xsl:if test="ancestor::db:itemizedlist
2951                or
2952                ancestor::db:orderedlist">
2953         <xsl:text>sub</xsl:text>

```

```

2954     </xsl:if>
2955     <xsl:text>paragraph*{</xsl:text>
2956     <xsl:apply-templates select="db:title/node()"/>
2957     <xsl:text>}&#xa;</xsl:text>
2958 </xsl:if>
2959 <xsl:if test="@xml:id">
2960     <xsl:value-of select="$armour"/>
2961     <xsl:text> \label{</xsl:text>
2962     <xsl:value-of select="@xml:id"/>
2963     <xsl:text>}&#xa;</xsl:text>
2964 </xsl:if>
2965 <xsl:value-of select="$armour"/>
2966 <xsl:text> \begin{enumerate</xsl:text>
2967 <xsl:if test="parent::db:para">
2968     <xsl:text>*</xsl:text>
2969 </xsl:if>
2970 <xsl:text>}</xsl:text>
2971 <!-- check for options needed -->
2972 <xsl:if test="parent::db:para
2973             or
2974             (@linkend and
2975             preceding::db:orderedlist
2976             [@xml:id=current()/@linkend])
2977             or
2978             @spacing='compact'
2979             or
2980             local-name()='simplelist'
2981             or
2982             @startingnumber
2983             or
2984             db:listitem/db:annotation">
2985     <xsl:text>[</xsl:text>
2986     <!-- inline -->
2987     <xsl:if test="parent::db:para">
2988         <xsl:text>label=\emph{\alph*}),</xsl:text>
2989         <xsl:text>itemjoin={}; },</xsl:text>
2990         <xsl:text>itemjoin*={}; and }</xsl:text>
2991     </xsl:if>
2992     <!-- resumption -->
2993     <xsl:if
2994         test="@linkend and
2995             preceding::db:orderedlist
2996             [@xml:id=current()/@linkend]">
2997         <xsl:if test="parent::db:para">
2998             <xsl:text>,</xsl:text>
2999         </xsl:if>
3000         <xsl:text>resume</xsl:text>
3001     </xsl:if>
3002     <!-- spacing -->
3003     <xsl:if test="@spacing='compact'
3004                 or
3005                 local-name()='simplelist'">
3006     <xsl:if
3007         test="parent::db:para or

```

```

3008         (@linkend and
3009         preceding::db:orderedlist
3010         [@xml:id=current()/@linkend]))">
3011     <xsl:text>,</xsl:text>
3012 </xsl:if>
3013     <xsl:text>noitemsep</xsl:text>
3014 </xsl:if>
3015 <!-- numbering -->
3016 <xsl:if test="@startingnumber">
3017     <xsl:if
3018         test="parent::db:para or
3019         (@linkend and
3020         preceding::db:orderedlist
3021         [@xml:id=current()/@linkend])
3022         or
3023         @spacing='compact'
3024         or
3025         local-name()='simplelist'">
3026         <xsl:text>,</xsl:text>
3027     </xsl:if>
3028     <xsl:text>start=</xsl:text>
3029     <xsl:value-of select="@startingnumber"/>
3030 </xsl:if>
3031 <!-- no indent if there's an annotation to come -->
3032 <xsl:if test="db:listitem/db:annotation">
3033     <xsl:if test="parent::db:para
3034         or
3035         (@linkend and
3036         preceding::db:orderedlist
3037         [@xml:id=current()/@linkend])
3038         or
3039         @spacing='compact'
3040         or
3041         local-name()='simplelist'
3042         or
3043         @startingnumber">
3044         <xsl:text>,</xsl:text>
3045     </xsl:if>
3046     <xsl:text>leftmargin=0.5em</xsl:text>
3047 </xsl:if>
3048     <xsl:text>]</xsl:text>
3049 </xsl:if>
3050 <!-- set indent if the list contains annotations -->
3051 <xsl:if test="db:listitem/db:annotation">
3052     <xsl:text>\setlength{\CPKannotationindent}{1em}</xsl:text>
3053 </xsl:if>
3054 <!-- do list -->
3055 <xsl:text>&#xa;</xsl:text>
3056 <xsl:apply-templates/>
3057 <!-- no withdrawal of armour needed
3058     for inline list termination
3059     because the end of a list item para
3060     will have created a newline -->
3061 <xsl:text>% \end{enumerate}</xsl:text>

```

```

3062     <xsl:if test="parent::db:para">
3063       <xsl:text>*</xsl:text>
3064     </xsl:if>
3065     <xsl:text>}</xsl:text>
3066     <!-- unset indent if the list contains annotations -->
3067     <xsl:if test="db:listitem/db:annotation">
3068       <xsl:text>\setlength{\CPKannotationindent}{0pt}</xsl:text>
3069     </xsl:if>
3070     <xsl:if test="not(parent::db:para)">
3071       <xsl:text>&#xa;</xsl:text>
3072     </xsl:if>
3073     <!-- check format -->
3074     <xsl:if test="@role='twocol'">
3075       <xsl:text>% \end{multicols}&#xa;</xsl:text>
3076     </xsl:if>
3077   </xsl:template>
3078
3079   <xsl:template
3080     match="db:orderedlist/db:title |
3081           db:procedure/db:title |
3082           db:simplelist[@arch='enumerate']/db:title"/>

```

Suppress any title when it is presented in document order.

db:variablelist Definition lists. Map exactly to the description environment.

```

3083   <xsl:template match="db:variablelist">
3084     <xsl:if test="db:title">
3085       <xsl:value-of select="$armour"/>
3086       <xsl:text>% \</xsl:text>
3087       <xsl:if
3088         test="ancestor::db:itemizedlist
3089              or
3090              ancestor::db:orderedlist">
3091         <xsl:text>sub</xsl:text>
3092       </xsl:if>
3093       <xsl:text>paragraph*{</xsl:text>
3094       <xsl:apply-templates select="db:title/node()"/>
3095       <xsl:text>}&#xa;</xsl:text>
3096     </xsl:if>
3097     <xsl:text>% \begin{description}[style=unboxed</xsl:text>
3098     <xsl:if test="@spacing='compact'">
3099       <xsl:text>,noitemsep</xsl:text>
3100     </xsl:if>
3101     <xsl:if test="db:varlistentry/db:listitem/db:annotation">
3102       <xsl:text>,leftmargin=0.5em</xsl:text>
3103     </xsl:if>
3104     <xsl:text>]</xsl:text>
3105     <!-- set indent if the list contains annotations -->
3106     <xsl:if test="db:varlistentry/db:listitem/db:annotation">
3107       <xsl:text>\setlength{\CPKannotationindent}{1em}</xsl:text>
3108     </xsl:if>
3109     <xsl:if test="@xml:id">
3110       <xsl:text>\label{</xsl:text>

```

```

3111     <xsl:value-of select="@xml:id"/>
3112     <xsl:text></xsl:text>
3113 </xsl:if>
3114     <xsl:text>&#xa;</xsl:text>
3115     <xsl:apply-templates/>
3116     <xsl:text>% \end{description}</xsl:text>
3117     <!-- unset indent if the list contains annotations -->
3118     <xsl:if test="db:varlistentry/db:listitem/db:annotation">
3119       <xsl:text>\setlength{\CPKannotationindent}{0pt}</xsl:text>
3120     </xsl:if>
3121     <xsl:text>&#xa;</xsl:text>
3122 </xsl:template>
3123
3124 <xsl:template match="db:variablelist/db:title"/>

```

db:simplelist [db:simplelist with [role='twocol'] and db:simplelist with [role] and db:member with [parent::db:simplelist[role]]]

Simple lists are bulleted lists with tight spacing and no bullet preset. The @role can specify two-column layout.

```

3125 <xsl:template match="db:simplelist[@role='twocol']">
3126   <xsl:text>% \begin{multicols}{2}</xsl:text>
3127   <xsl:if test="@xreflabel">
3128     <xsl:text>[\paragraph*{</xsl:text>
3129     <xsl:value-of select="normalize-space(@xreflabel)"/>
3130     <xsl:text>}]</xsl:text>
3131   </xsl:if>
3132   <xsl:text>\begin{itemize}[noitemsep]&#xa;</xsl:text>
3133   <xsl:apply-templates/>
3134   <xsl:text>% \end{itemize}
3135 % \end{multicols}&#xa;</xsl:text>
3136 </xsl:template>
3137
3138 <xsl:template match="db:simplelist[not(@role)
3139   and
3140   not(@arch)]">
3141   <xsl:text>% \begin{itemize}</xsl:text>
3142   <xsl:text>[label={},noitemsep]&#xa;</xsl:text>
3143   <xsl:apply-templates/>
3144   <xsl:text>% \end{itemize}&#xa;</xsl:text>
3145 </xsl:template>
3146
3147 <xsl:template match="db:simplelist[@role]">
3148   <xsl:text>&#xa;% \</xsl:text>
3149   <xsl:value-of select="@role"/>
3150   <xsl:if test="@remap='longest'">
3151     <xsl:text>\longestline{</xsl:text>
3152     <xsl:value-of
3153       select="normalize-space(db:member
3154         [not(..db:member/string-length()
3155           >string-length(.))][1])"/>
3156     <xsl:text>}</xsl:text>
3157   </xsl:if>

```

```

3158     <xsl:text>\stanza&#xa;</xsl:text>
3159     <xsl:apply-templates/>
3160 </xsl:template>
3161
3162 <xsl:template
3163     match="db:member[parent::db:simplelist[@role]]">
3164     <xsl:text>% </xsl:text>
3165     <xsl:apply-templates/>
3166     <xsl:if test="position()=last()">
3167         <xsl:text>\</xsl:text>
3168     </xsl:if>
3169     <xsl:text>&#xa;</xsl:text>
3170 </xsl:template>

```

db:listitem [in db:itemizedlist/ and db:orderedlist/db:listitem and
db:procedure/db:step and db:simplelist with [not(@role)]/db:member]

Items allow for variation of the bullet or number, and for a label from the
@xml:id attribute. This template does not match for <variablelist> elements,
see the term template below.

```

3171 <xsl:template
3172     match="db:itemizedlist/db:listitem |
3173           db:orderedlist/db:listitem |
3174           db:procedure/db:step |
3175           db:simplelist[not(@role)]/db:member">
3176     <xsl:text>% \item</xsl:text>
3177     <xsl:choose>
3178         <!-- remap a bullet or number -->
3179         <xsl:when test="@remap or parent::db:simplelist">
3180             <xsl:text>[</xsl:text>
3181             <xsl:value-of select="@remap"/>
3182             <xsl:text>]</xsl:text>
3183         </xsl:when>
3184         <!-- or prefix the number with a symbol
3185              (what was this for?!) -->
3186         <xsl:when test="@role">
3187             <xsl:text>\leavevmode\llap{</xsl:text>
3188             <xsl:value-of select="@role"/>
3189             <xsl:text>\thinspace}</xsl:text>
3190         </xsl:when>
3191         <!-- normally just a space -->
3192         <xsl:otherwise>
3193             <xsl:text> </xsl:text>
3194         </xsl:otherwise>
3195     </xsl:choose>
3196     <!-- label it if needed -->
3197     <xsl:if test="@xml:id">
3198         <xsl:text>\label{</xsl:text>
3199         <xsl:value-of select="@xml:id"/>
3200         <xsl:text>}</xsl:text>
3201     </xsl:if>
3202     <xsl:apply-templates/>
3203     <!-- syntactic punctuation now handled by db:para template

```

```

3204         except for db:member, which needs terminating -->
3205         <xsl:if test="local-name()='member'">
3206             <xsl:text>&#xa;</xsl:text>
3207         </xsl:if>
3208     </xsl:template>

```

`db:term` The `<term>` is the label of a definition or description list, so in \LaTeX it needs to usurp the `\item` command and place its content into what is normally the optional argument to `\item`. The `listitem` *template* (above) does not match the `<listitem>` child *elements* of a `<varlistentry>` element for this reason.

```

3209     <xsl:template match="db:term">
3210         <xsl:if
3211             test="ancestor::db:part[@xml:id='code']
3212                 and ancestor::db:chapter/@xlink:href
3213                 and ancestor::db:chapter/@xml:id
3214                 and not(substring-after(
3215                     ancestor::db:chapter/@xlink:href, '.')='dtd'
3216                     or substring-after(
3217                         ancestor::db:chapter/@xlink:href, '.')='xml'
3218                     or substring-after(
3219                         ancestor::db:chapter/@xlink:href, '.')='xsl')'">
3220             <xsl:text>% \iffalse&#xa;</xsl:text>
3221             <xsl:value-of select="$sentinel"/>
3222             <xsl:text>&#xa;</xsl:text>
3223             <xsl:apply-templates select="." mode="readme"/>
3224             <xsl:text>% \fi&#xa;</xsl:text>
3225         </xsl:if>
3226         <xsl:text>% \item</xsl:text>
3227         <xsl:if test="@annotations">
3228             <xsl:text>\llap{</xsl:text>
3229             <xsl:value-of select="@annotations"/>
3230             <xsl:text>\ }</xsl:text>
3231         </xsl:if>
3232         <xsl:if test="ancestor::db:variablelist[@role='numbered']">
3233             <xsl:number count="db:varlistentry" format="1"/>
3234             <xsl:text>.</xsl:text>
3235         </xsl:if>
3236         <xsl:apply-templates/>
3237         <xsl:if test="not(db:command[@conformance or @condition])">
3238             <xsl:text>:</xsl:text>
3239         </xsl:if>
3240         <xsl:text>]</xsl:text>
3241         <xsl:if test="following-sibling::db:listitem/db:annotation">
3242             <xsl:text>\par </xsl:text>
3243         </xsl:if>
3244         <xsl:if test="../@xml:id">
3245             <xsl:text>\label{</xsl:text>
3246             <xsl:value-of select="../@xml:id"/>
3247             <xsl:text>}</xsl:text>
3248         </xsl:if>
3249     </xsl:template>
3250

```

```

3251 <xsl:template match="db:termdef">
3252   <xsl:text>\textbf{</xsl:text>
3253   <xsl:apply-templates/>
3254   <xsl:text>}\index{</xsl:text>
3255   <xsl:value-of select="normalize-space(.)"/>
3256   <xsl:text>|textbf{</xsl:text>
3257 </xsl:template>

```

db:formalpara In normal text, this uses the `<title>` subelement as a heading. However, in the `<abstract>`, (where lists are not allowed by *DocBook*), it gets abused/transformed to a one-item bulleted list with vertical spacing suppressed.

```

3258 <xsl:template match="db:formalpara[not(parent::db:abstract)]">
3259   <xsl:apply-templates/>
3260 </xsl:template>
3261
3262 <xsl:template match="db:formalpara/db:title">
3263   <xsl:text>% \par\textbf{</xsl:text>
3264   <xsl:apply-templates/>
3265   <xsl:text>}\par\noindent\ignorespaces </xsl:text>
3266 </xsl:template>
3267
3268 <xsl:template match="db:formalpara[parent::db:abstract]">
3269   <xsl:text>% \begin{itemize}[nosep]&#xa;% \item </xsl:text>
3270   <xsl:apply-templates/>
3271   <xsl:text>% \end{itemize}&#xa;</xsl:text>
3272 </xsl:template>

```

11.4.5 Flow

These templates all match element types that occur *within* running text (eg inside paragraphs).

Because of the design flaw in XSL in removing white-space nodes in mixed content *even when a DTD/schema is in use*, almost all these templates start with the detection of an immediately-preceding element, and in most cases compensate for the missing white-space.

db:acronym If there is an `@xml:id` attribute present, this is taken as a defining instance, with the content being the expansion, so it is formatted as the expansion first and the acronym in parentheses afterwards, and indexed as such.

When there is no `@xml:id` attribute present, it is assumed that the content is the acronym, and formatted accordingly and added to the index.

```

3273 <xsl:template match="db:acronym">
3274   <xsl:call-template name="compensate-space"/>
3275   <xsl:choose>
3276     <!-- ID present: defining instance -->
3277     <xsl:when test="@xml:id">

```

```

3278      <!-- output the content (the expansion),
3279             quoted if required -->
3280      <xsl:if test="@remap='quoted'">
3281        <xsl:text>'</xsl:text>
3282      </xsl:if>
3283      <xsl:apply-templates/>
3284      <xsl:text>\phantomsection\label{</xsl:text>
3285      <xsl:value-of select="@xml:id"/>
3286      <xsl:text>}</xsl:text>
3287      <xsl:if test="not(ancestor::db:table
3288                    or ancestor::db:informaltable
3289                    or ancestor::db:figure)">
3290        <xsl:text>\marginpar{\small\HandRight}</xsl:text>
3291      </xsl:if>
3292      <!-- output any plural specified in @role -->
3293      <xsl:if test="@role">
3294        <xsl:value-of select="@role"/>
3295      </xsl:if>
3296      <!-- end quote -->
3297      <xsl:if test="@remap='quoted'">
3298        <xsl:text>'</xsl:text>
3299      </xsl:if>
3300      <!-- output the acronym in parentheses -->
3301      <xsl:text> (</xsl:text>
3302      <xsl:choose>
3303        <!-- use a logo -->
3304        <xsl:when test="@remap">
3305          <xsl:value-of select="@remap"/>
3306        </xsl:when>
3307        <xsl:otherwise>
3308          <xsl:call-template name="casestyle">
3309            <xsl:with-param name="text"
3310              select="normalize-space(@xml:id)"/>
3311          </xsl:call-template>
3312          <!-- pluralise the acronym too if needed -->
3313          <xsl:choose>
3314            <xsl:when test="@role">
3315              <xsl:value-of select="@role"/>
3316            </xsl:when>
3317            <xsl:when test="@arch='plural'">
3318              <xsl:text>s</xsl:text>
3319            </xsl:when>
3320          </xsl:choose>
3321        </xsl:otherwise>
3322      </xsl:choose>
3323      <!-- index it as the locus classicus
3324           NOT index the acro, see expansion
3325           BUT index expansion, see acro
3326 -->
3327      <xsl:text>)\index{</xsl:text>
3328      <xsl:value-of select="normalize-space(.)"/>
3329      <xsl:text>|see{</xsl:text>
3330      <xsl:value-of select="upper-case(@xml:id)"/>
3331      <xsl:text>}}\index{</xsl:text>

```

```

3332     <xsl:value-of select="upper-case(@xml:id)"/>
3333     <xsl:text>|textbf}</xsl:text>
3334 </xsl:when>
3335 <!-- no ID: normal use -->
3336 <xsl:otherwise>
3337     <!-- output the acronym -->
3338     <xsl:variable name="definst"
3339       select="//db:acronym[@xml:id=current()/.]"/>
3340     <xsl:choose>
3341       <!-- logo? -->
3342       <xsl:when test="$definst and @remap">
3343         <xsl:value-of
3344           select="//db:acronym[@xml:id=current()/.]
3345             /@remap"/>
3346         <xsl:text>{}</xsl:text>
3347       </xsl:when>
3348       <!-- not in the same chapter, and never within the
3349         term argument to a description list -->
3350       <xsl:when test="$definst
3351         and
3352         count($definst/ancestor::db:chapter
3353           |
3354           ./ancestor::db:chapter)>1
3355         and not(ancestor::db:term)">
3356         <xsl:text>\hyperref[</xsl:text>
3357         <xsl:value-of select="."/>
3358         <xsl:text>]</xsl:text>
3359         <xsl:call-template name="casestyle">
3360           <xsl:with-param name="text"
3361             select="normalize-space(.)"/>
3362         </xsl:call-template>
3363         <xsl:text>}</xsl:text>
3364       <!-- pluralised if necessary -->
3365       <xsl:choose>
3366         <xsl:when test="@role">
3367           <xsl:value-of select="@role"/>
3368         </xsl:when>
3369         <xsl:when test="@arch='plural'">
3370           <xsl:text>s</xsl:text>
3371         </xsl:when>
3372       </xsl:choose>
3373     </xsl:when>
3374     <xsl:otherwise>
3375       <xsl:call-template name="casestyle">
3376         <xsl:with-param name="text"
3377           select="normalize-space(.)"/>
3378       </xsl:call-template>
3379       <!-- pluralised if necessary -->
3380       <xsl:if test="@role">
3381         <xsl:value-of select="@role"/>
3382       </xsl:if>
3383     </xsl:otherwise>
3384   </xsl:choose>
3385   <!-- index it -->

```

```

3386         <xsl:if test="//db:acronym[@xml:id=current()/.]">
3387             <xsl:if test="ancestor::db:footnote">
3388                 <xsl:text>\protect</xsl:text>
3389             </xsl:if>
3390             <xsl:text>\index{</xsl:text>
3391             <xsl:value-of select="normalize-space(.)"/>
3392             <xsl:text>}</xsl:text>
3393         </xsl:if>
3394     </xsl:otherwise>
3395 </xsl:choose>
3396 </xsl:template>

```

`db:citetitle` Cites the title of a bibliographic reference. If there is no `@linkend` IDREF, or if the reference is to a book, the title is italicised, otherwise it goes in quotes. If the link is present, the title text is taken from the bibliographic entry and the element content is ignored. The title is followed by a citation in parentheses.

```

3397 <xsl:template match="db:citetitle">
3398     <xsl:call-template name="compensate-space"/>
3399     <xsl:choose>
3400         <xsl:when
3401             test="not(@linkend) or
3402                 //*[@xml:id=current()/@linkend]/@areflabel='book'">
3403             <xsl:text>\emph{</xsl:text>
3404         </xsl:when>
3405         <xsl:otherwise>
3406             <xsl:text>"</xsl:text>
3407         </xsl:otherwise>
3408     </xsl:choose>
3409     <xsl:choose>
3410         <xsl:when test="@linkend">
3411             <xsl:apply-templates
3412                 select="//*[@xml:id=current()/@linkend]/db:title/node()"/>
3413         </xsl:when>
3414         <xsl:otherwise>
3415             <xsl:apply-templates/>
3416         </xsl:otherwise>
3417     </xsl:choose>
3418     <xsl:choose>
3419         <xsl:when
3420             test="not(@linkend) or
3421                 //*[@xml:id=current()/@linkend]/@areflabel='book'">
3422             <xsl:text>}</xsl:text>
3423         </xsl:when>
3424         <xsl:otherwise>
3425             <xsl:text>"</xsl:text>
3426         </xsl:otherwise>
3427     </xsl:choose>
3428     <xsl:if test="//db:biblioentry[@xml:id=current()/@linkend]">
3429         <xsl:text> \parencite{</xsl:text>
3430         <xsl:value-of
3431             select="translate(normalize-space(@linkend), ' ', ',')"/>
3432         <xsl:text>}</xsl:text>

```

```

3433     </xsl:if>
3434 </xsl:template>

```

db:code Inline words of program code. Specified languages use L^AT_EX's `\verb` command; others are just set in monospace type.

```

3435 <xsl:template match="db:code">
3436   <xsl:call-template name="compensate-space"/>
3437   <xsl:choose>
3438     <xsl:when test="@language='RE'">
3439       <xsl:text>\verb`</xsl:text>
3440       <xsl:value-of select="."/>
3441       <xsl:text>/`</xsl:text>
3442     </xsl:when>
3443     <xsl:when test="@language='Cocoon'">
3444       <xsl:text>\verb`{</xsl:text>
3445       <xsl:value-of select="."/>
3446       <xsl:text>}</xsl:text>
3447     </xsl:when>
3448     <xsl:when test="@language='bash'">
3449       <xsl:text>\verb`</xsl:text>
3450       <xsl:value-of select="."/>
3451       <xsl:text>`</xsl:text>
3452     </xsl:when>
3453     <xsl:otherwise>
3454       <xsl:text>\texttt{</xsl:text>
3455       <xsl:apply-templates/>
3456       <xsl:text>}</xsl:text>
3457     </xsl:otherwise>
3458   </xsl:choose>
3459 </xsl:template>

```

db:classname This element type is used to identify a L^AT_EX document class, generated using the `\DescribeClass` command. Its use is limited to certain contexts in the "code" part (not in footnotes, titles, or terms; not when it is already the current annotation name; and not twice in the same context). An example of usage can be set up by using the `@condition` attribute to hold a sample optional argument, and the `@conformance` attribute to hold the package name mnemonic, eg `euroflag[<option>]{<packagename>}`

```

3460 <xsl:template match="db:classname">
3461   <xsl:call-template name="compensate-space"/>
3462   <xsl:choose>
3463     <xsl:when
3464       test="ancestor::db:part[@xml:id='code'] and
3465             not(ancestor::db:footnote) and
3466             not(ancestor::db:title) and
3467             not(ancestor::db:term) and
3468             not(.=ancestor::db:annotation/@xreflabel) and
3469             (@language='TeX'
3470              or @language='LaTeXe'
3471              or @language='LaTeX'
3472              or not(@language))

```

```

3473         and not(preceding-sibling::db:classname
3474                 [=current()/.]")">
3475     <xsl:text>\DescribeClass{</xsl:text>
3476     <xsl:value-of select="."/"/>
3477     <xsl:text>}</xsl:text>
3478 </xsl:when>
3479 <xsl:otherwise>
3480     <xsl:text>\index{</xsl:text>
3481     <xsl:value-of select="normalize-space(.)"/>
3482     <xsl:text>@\textsf{</xsl:text>
3483     <xsl:value-of select="normalize-space(.)"/>
3484     <xsl:text>} (class)</xsl:text>
3485     <xsl:text>}</xsl:text>
3486     <xsl:text>\index{classes:!!</xsl:text>
3487     <xsl:value-of select="normalize-space(.)"/>
3488     <xsl:text>@\textsf{</xsl:text>
3489     <xsl:value-of select="normalize-space(.)"/>
3490     <xsl:text>}}</xsl:text>
3491 </xsl:otherwise>
3492 </xsl:choose>
3493 <!-- always sans -->
3494 <xsl:text>\textsf{</xsl:text>
3495 <xsl:apply-templates/>
3496 <xsl:text>}</xsl:text>
3497 <xsl:if test="@condition">
3498     <xsl:text>\oarg{</xsl:text>
3499     <xsl:value-of select="normalize-space(@condition)"/>
3500     <xsl:text>}</xsl:text>
3501 </xsl:if>
3502 <xsl:if test="@conformance">
3503     <xsl:text>\marg{</xsl:text>
3504     <xsl:value-of select="normalize-space(@conformance)"/>
3505     <xsl:text>}</xsl:text>
3506 </xsl:if>
3507 </xsl:template>

```

db:command This identifies a command in various syntaxes. A command occurring in documentation uses the `\DescribeMacro`, or whatever other predefined equivalent is given in the `@remap` attribute (eg "counter", "switch", "length", etc: see `prepost.xml`). The `@role` attribute specifies that the command does not belong to the current package and is only mentioned incidentally, and therefore requires no indexing.

The backslash is added programmatically: it must not be typed — it gets faked up when in `<term>` (definition list) or when being used as an example (not for indexing); otherwise it uses `\verb` if needed. The named template `avoidverb` is called to ensure a delimiter is chosen that does not interfere with the content.

```

3508 <xsl:template match="db:command">
3509     <xsl:call-template name="compensate-space"/>
3510     <xsl:if
3511         test="@role and not(.='TeX') and not(.='LaTeX') and
3512         not(ancestor::db:term) and

```

```

3513         not(.=ancestor::db:annotation/@xreflabel) and
3514         (@language='TeX'
3515         or @language='LaTeXe'
3516         or @language='LaTeX'
3517         or not(@language))
3518         and not(preceding-sibling::db:command[.=current()/.]")">
3519 <xsl:text>\DescribeMacro{\</xsl:text>
3520 <xsl:choose>
3521   <xsl:when
3522     test="@language='TeX' or
3523           @language='LaTeXe' or
3524           @language='LaTeX' or
3525           not(@language)
3526           and contains(.,'{')">
3527     <xsl:value-of
3528       select="substring-before(normalize-space(.),'{'")"/>
3529   </xsl:when>
3530   <xsl:otherwise>
3531     <xsl:value-of select="normalize-space(.)"/>
3532   </xsl:otherwise>
3533 </xsl:choose>
3534 <xsl:text>}\</xsl:text>
3535 </xsl:if>
3536 <xsl:choose>
3537   <xsl:when
3538     test="(@language='TeX' or
3539           @language='LaTeXe' or
3540           @language='LaTeX' or
3541           not(@language))
3542           and (not(@role) or parent::db:term)">
3543     <xsl:if test="parent::db:term and
3544               @userlevel!='optional'">
3545       <xsl:text>\llap{$\star$\enspace}</xsl:text>
3546     </xsl:if>
3547   <xsl:choose>
3548     <xsl:when test="parent::db:term">
3549       <xsl:text>\texttt{\textbackslash{}</xsl:text>
3550       <xsl:apply-templates/>
3551       <xsl:text>}</xsl:text>
3552     </xsl:when>
3553     <xsl:otherwise>
3554       <xsl:call-template name="avoidverb"/>
3555     </xsl:otherwise>
3556   </xsl:choose>
3557 </xsl:when>
3558 <!-- use \verb for TeX commands if needed -->
3559 <xsl:when
3560   test="@language='TeX' or
3561         @language='LaTeXe' or
3562         @language='LaTeX' or
3563         not(@language)">
3564   <xsl:call-template name="avoidverb"/>
3565 </xsl:when>
3566 <!-- bash -->

```

```

3567     <xsl:when test="@language='bash'">
3568         <xsl:call-template name="avoidverb"/>
3569         <xsl:if test="@userlevel">
3570             <xsl:text>\thinspace(</xsl:text>
3571             <xsl:value-of select="@userlevel"/>
3572             <xsl:text>)</xsl:text>
3573         </xsl:if>
3574     </xsl:when>
3575     <!-- other language commands have no backslash -->
3576     <xsl:when test="@condition='nolit'">
3577         <xsl:call-template name="avoidverb"/>
3578     </xsl:when>
3579     <xsl:otherwise>
3580         <xsl:call-template name="avoidverb"/>
3581     </xsl:otherwise>
3582 </xsl:choose>
3583 <xsl:if test="@condition and
3584             @condition!='nolit'">
3585     <xsl:text>\oarg{</xsl:text>
3586     <xsl:value-of
3587         select="normalize-space(@condition)"/>
3588     <xsl:text>}</xsl:text>
3589 </xsl:if>
3590 <xsl:if test="@conformance">
3591     <xsl:text>\marg{</xsl:text>
3592     <xsl:value-of
3593         select="normalize-space(@conformance)"/>
3594     <xsl:text>}</xsl:text>
3595 </xsl:if>
3596 </xsl:template>

```

db:emphasis Simple italicisation by default, or bold when the @role attribute is "strong".

```

3597 <xsl:template match="db:emphasis">
3598     <xsl:call-template name="compensate-space"/>
3599     <xsl:choose>
3600         <xsl:when test="@role='strong'">
3601             <xsl:text>\textbf{</xsl:text>
3602         </xsl:when>
3603         <xsl:when test="ancestor::db:emphasis">
3604             <xsl:text>\textbf{\upshape </xsl:text>
3605         </xsl:when>
3606         <xsl:otherwise>
3607             <xsl:text>\emph{</xsl:text>
3608         </xsl:otherwise>
3609     </xsl:choose>
3610     <xsl:apply-templates/>
3611     <xsl:text>}</xsl:text>
3612 </xsl:template>

```

db:email This occurs in the metadata for the title page and just needs formatting as a URI.

```

3613 <xsl:template match="db:email">
3614     <xsl:text>\url{</xsl:text>

```

```

3615     <xsl:value-of select="."/>
3616     <xsl:text>}</xsl:text>
3617 </xsl:template>

```

`db:envvar` This element is used to identify a \LaTeX environment. As with the `<command>` element, the `@role` attribute specifies the command does not belong to this package or class and is only mentioned incidentally. And as with `<classname>`, the `@condition` and `@conformance` attributes are used to add metadata for describing the example.

```

3618 <xsl:template match="db:envvar">
3619   <xsl:call-template name="compensate-space"/>
3620   <xsl:if test="@role and
3621               not(ancestor::db:term) and
3622               not(.=ancestor::db:annotation/@xreflabel) and
3623               (@language='TeX'
3624               or @language='LaTeXe'
3625               or @language='LaTeX'
3626               or not(@language))
3627               and not(preceding-sibling::db:envvar
3628                       [.=current()/])">
3629     <xsl:text>\DescribeEnv{</xsl:text>
3630     <xsl:apply-templates/>
3631     <xsl:text>}</xsl:text>
3632   </xsl:if>
3633   <xsl:text>\texttt{</xsl:text>
3634   <xsl:apply-templates/>
3635   <xsl:text>}</xsl:text>
3636   <xsl:if test="@condition">
3637     <xsl:text>\oarg{</xsl:text>
3638     <xsl:value-of select="normalize-space(@condition)"/>
3639     <xsl:text>}</xsl:text>
3640   </xsl:if>
3641   <xsl:if test="@conformance">
3642     <xsl:text>\marg{</xsl:text>
3643     <xsl:value-of select="normalize-space(@conformance)"/>
3644     <xsl:text>}</xsl:text>
3645   </xsl:if>
3646 </xsl:template>

```

`db:errortext` Formal error text (error messages) can be identified and will be reproduced and indexed using DocTeX.

```

3647 <xsl:template match="db:errortext">
3648   <xsl:call-template name="compensate-space"/>
3649   <xsl:text>\DescribeError{</xsl:text>
3650   <xsl:value-of select="."/>
3651   <xsl:text>}\textsf{</xsl:text>
3652   <xsl:value-of select="."/>
3653   <xsl:text>}</xsl:text>
3654 </xsl:template>

```

`db:exceptionname` This element is used to identify the action words of RFC 2119 (MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL). Its use is detected in processing and the warning text automatically included in the preamble to the documentation.

```

3655 <xsl:template match="db:exceptionname">
3656   <xsl:call-template name="compensate-space"/>
3657   <xsl:text>{\sffamily </xsl:text>
3658   <xsl:call-template name="casestyle">
3659     <xsl:with-param name="text" select="normalize-space(.)"/>
3660   </xsl:call-template>
3661   <xsl:text>}</xsl:text>
3662 </xsl:template>

```

`db:filename` [`db:filename` and `db:systemitem`]

`avoidverb` Both share the same model and use the `avoidverb` named template to format in the monospace font.

```

3663 <xsl:template match="db:filename | db:systemitem">
3664   <xsl:call-template name="compensate-space"/>
3665   <xsl:call-template name="avoidverb"/>
3666   <xsl:if
3667     test="contains('.,;:?!',
3668       substring(following-sibling::node()[1],1,1))">
3669     <xsl:text>\thinspace</xsl:text>
3670   </xsl:if>
3671 </xsl:template>

```

`db:firstterm` Used to identify the first occurrence of a specialist term that need explanation (bold italics).

```

3672 <xsl:template match="db:firstterm">
3673   <xsl:call-template name="compensate-space"/>
3674   <xsl:text>\textbf{\emph{</xsl:text>
3675   <xsl:apply-templates/>
3676   <xsl:text>}}\index{</xsl:text>
3677   <xsl:value-of select="normalize-space(.)"/>
3678   <xsl:text>|textbf}</xsl:text>
3679 </xsl:template>

```

`db:footnote` Identified footnotes.

```

3680 <xsl:template match="db:footnote">
3681   <xsl:text>\footnote{&#xa;</xsl:text>
3682   <xsl:apply-templates/>
3683   <xsl:text>}</xsl:text>
3684 </xsl:template>

```

`db:foreignphrase` Identifies a word or phrase in a language other than the base language of the document (italicised, but cjk is recognised; others use the `\foreignlanguage` command of the babel package).

```

3685 <xsl:template match="db:foreignphrase">
3686   <xsl:call-template name="compensate-space"/>
3687   <xsl:choose>
3688     <xsl:when test="@xml:lang='jp'">
3689       <xsl:text>\cjktext{</xsl:text>
3690       <xsl:apply-templates/>
3691       <xsl:text>}</xsl:text>
3692     </xsl:when>
3693     <xsl:when test="@xml:lang='ga'">
3694       <!-- <xsl:text>\texteiad{</xsl:text>-->
3695       <xsl:text>{</xsl:text>
3696       <xsl:value-of select="@remap"/>
3697       <xsl:apply-templates/>
3698       <xsl:text>}</xsl:text>
3699     </xsl:when>
3700     <xsl:when test="@xml:lang!=/db:book/@xml:lang">
3701       <xsl:text>\foreignlanguage{</xsl:text>
3702       <xsl:value-of
3703         select="$langs/language[concat(@Lang,'-',@Country)
3704           =current()/@xml:lang]/@babel"/>
3705       <xsl:text>}{</xsl:text>
3706     <xsl:choose>
3707       <xsl:when test="$langs/language
3708         [@iso=current()/@xml:lang]
3709         /@wsys='la'">
3710         <xsl:text>\emph{</xsl:text>
3711         <xsl:apply-templates/>
3712         <xsl:text>}</xsl:text>
3713       </xsl:when>
3714       <xsl:otherwise>
3715         <xsl:apply-templates/>
3716       </xsl:otherwise>
3717     </xsl:choose>
3718     <xsl:text>}</xsl:text>
3719   </xsl:when>
3720   <xsl:otherwise>
3721     <xsl:text>\emph{</xsl:text>
3722     <xsl:apply-templates/>
3723     <xsl:text>}</xsl:text>
3724   </xsl:otherwise>
3725 </xsl:choose>
3726 </xsl:template>

```

db:function Identifies computer language functions, including xSLT templates. Without the **@role** and **@language** attributes, the function is formatted and indexed with the **\DescribeFunction** command.

```

3727 <xsl:template match="db:function">
3728   <xsl:call-template name="compensate-space"/>
3729   <xsl:if test="not(@role) and not(@language)">
3730     <xsl:text>\DescribeFunction{</xsl:text>
3731     <xsl:value-of select="."/>
3732     <xsl:text>}</xsl:text>
3733   </xsl:if>

```

```

3734     <xsl:text>\texttt{</xsl:text>
3735     <xsl:value-of select="."/>
3736     <xsl:if test="@language='XSLT'">
3737         <xsl:text>(</xsl:text>
3738     </xsl:if>
3739     <xsl:text>}</xsl:text>
3740 </xsl:template>

```

db:guimenu Identifies a named menu in a GUI. Menu formatting is done with the menukeys package; adjacent elements in the <gui*> group are formatted together.

```

3741 <!-- gui menu stuff -->
3742 <xsl:template match="db:guimenu">
3743     <xsl:call-template name="compensate-space"/>
3744     <xsl:choose>
3745         <xsl:when
3746             test="starts-with(local-name(
3747                 preceding-sibling::node()[1]),'gui')">
3748             <xsl:message>
3749                 <xsl:text>ERROR: top-level guimenu "</xsl:text>
3750                 <xsl:value-of select="."/>
3751                 <xsl:text>" preceded by </xsl:text>
3752                 <xsl:value-of select="name(preceding-sibling::*[1])"/>
3753             </xsl:message>
3754         </xsl:when>
3755         <xsl:otherwise>
3756             <xsl:text>\menu{</xsl:text>
3757         </xsl:otherwise>
3758     </xsl:choose>
3759     <xsl:apply-templates/>
3760     <xsl:if
3761         test="not(starts-with(local-name(
3762             following-sibling::node()[1]),'gui'))">
3763         <xsl:text>}</xsl:text>
3764     </xsl:if>
3765 </xsl:template>

```

db:guisubmenu Identifies a named submenu in a GUI. See note above.

```

3766 <xsl:template match="db:guisubmenu">
3767     <xsl:choose>
3768         <xsl:when
3769             test="starts-with(local-name(
3770                 preceding-sibling::node()[1]),'gui')">
3771             <xsl:text>></xsl:text>
3772         </xsl:when>
3773         <xsl:otherwise>
3774             <xsl:text>\menu{</xsl:text>
3775         </xsl:otherwise>
3776     </xsl:choose>
3777     <xsl:apply-templates/>
3778     <xsl:if
3779         test="not(starts-with(local-name(

```

```

3780         following-sibling::node()[1]),'gui'))">
3781     <xsl:text>}</xsl:text>
3782 </xsl:if>
3783 </xsl:template>

```

db:guimenuitem Identifies a named menu item in a GUI. See note above.

```

3784 <xsl:template match="db:guimenuitem">
3785     <xsl:choose>
3786         <xsl:when
3787             test="starts-with(local-name(
3788                 preceding-sibling::node()[1]),'gui'))">
3789             <xsl:text></xsl:text>
3790         </xsl:when>
3791         <xsl:otherwise>
3792             <xsl:text>\menu{</xsl:text>
3793         </xsl:otherwise>
3794     </xsl:choose>
3795     <xsl:apply-templates/>
3796     <xsl:if
3797         test="not(starts-with(local-name(
3798             following-sibling::node()[1]),'gui'))">
3799         <xsl:text>}</xsl:text>
3800     </xsl:if>
3801 </xsl:template>

```

db:guiicon Uses the *FontAwesome* icon if one is named in the @remap attribute, otherwise uses the @annotations (for options) and @xlink:href (for filename) attributes to construct an \includegraphics command.

```

3802 <xsl:template match="db:guiicon">
3803     <xsl:call-template name="compensate-space"/>
3804     <xsl:choose>
3805         <!-- use FontAwesome icon -->
3806         <xsl:when test="@remap">
3807             <xsl:text>\fa</xsl:text>
3808             <xsl:value-of select="@remap"/>
3809             <xsl:text>{</xsl:text>
3810         </xsl:when>
3811         <!-- otherwise supplied image -->
3812         <xsl:otherwise>
3813             <xsl:text>\includegraphics</xsl:text>
3814             <xsl:if test="@annotations">
3815                 <xsl:text>[</xsl:text>
3816                 <xsl:value-of select="@annotations"/>
3817                 <xsl:text>]</xsl:text>
3818             </xsl:if>
3819             <xsl:text>{</xsl:text>
3820             <xsl:value-of select="@xlink:href"/>
3821             <xsl:text>}</xsl:text>
3822         </xsl:otherwise>
3823     </xsl:choose>
3824     <!-- element is normally empty -->

```

```

3825     <xsl:apply-templates/>
3826 </xsl:template>

```

db:guilabel Uses the angular keys of the menukeys package to represent icons in a GUI.

```

3827 <xsl:template match="db:guilabel">
3828   <xsl:call-template name="compensate-space"/>
3829   <xsl:text>{\renewmenu macro{\keys}{angularkeys}\keys{</xsl:text>
3830   <xsl:apply-templates/>
3831   <xsl:text>}}</xsl:text>
3832 </xsl:template>

```

db:guibutton Uses the \changemenu color of the menukeys package to represent on-screen buttons in a GUI.

```

3833 <xsl:template match="db:guibutton">
3834   <xsl:call-template name="compensate-space"/>
3835   <xsl:text>{\changemenu color{\keys}{bg}%</xsl:text>
3836   <xsl:text> {HTML}{CDCDCD}\renewmenu macro{\keys}%</xsl:text>
3837   <xsl:text> {shadowed angularkeys}\keys{</xsl:text>
3838   <xsl:apply-templates/>
3839   <xsl:text>}}</xsl:text>
3840 </xsl:template>

```

db:important Identifies a segment of text of high importance and formats it in bold sans-serif type.

```

3841 <xsl:template match="db:important">
3842   <xsl:text>\begingroup\sffamily\bfseries&#xa;</xsl:text>
3843   <xsl:apply-templates/>
3844   <xsl:text>% \endgroup </xsl:text>
3845 </xsl:template>

```

db:indexterm Identifies a word or phrase needing indexing. With an @xml:id attribute, it puts the page number in bold in the index.

```

3846 <xsl:template match="db:indexterm">
3847   <xsl:text>% \index{</xsl:text>
3848   <xsl:apply-templates/>
3849   <xsl:choose>
3850     <xsl:when test="@xml:id and not(db:see)">
3851       <xsl:text>|textbf</xsl:text>
3852     </xsl:when>
3853   </xsl:choose>
3854   <xsl:text>}&#xa;</xsl:text>
3855 </xsl:template>

```

db:keycap Identifies a single key cap on the keyboard for the implication that the user presses it.

```

3856 <xsl:template match="db:keycap">
3857   <xsl:call-template name="compensate-space"/>

```

```

3858     <xsl:text>\keys{</xsl:text>
3859     <xsl:choose>
3860       <xsl:when test="contains('$%&#38;#',',.')">
3861         <xsl:text>\</xsl:text>
3862         <xsl:apply-templates/>
3863       </xsl:when>
3864       <xsl:when test=".='^' or .= '~'">
3865         <xsl:text>\</xsl:text>
3866         <xsl:apply-templates/>
3867         <xsl:text>{}</xsl:text>
3868       </xsl:when>
3869       <xsl:when test="contains('{&lt;|&gt;}','/')">
3870         <xsl:text>\[</xsl:text>
3871         <xsl:apply-templates/>
3872         <xsl:text>\]</xsl:text>
3873       </xsl:when>
3874       <xsl:when test=".='\''">
3875         <xsl:text>\textbackslash</xsl:text>
3876       </xsl:when>
3877       <xsl:otherwise>
3878         <xsl:value-of select="."/>
3879       </xsl:otherwise>
3880     </xsl:choose>
3881     <xsl:text>}</xsl:text>
3882   </xsl:template>

```

db:link Identifies a link to an external resource, typically a URI. If the `hyperref` package has been requested for the documentation, use the `\hyperlink` command for the `@xlink:href` attribute, otherwise use `\url` for the `@xlink:href` attribute; but if the content is non-null, just make it blue and underlined.

```

3883   <xsl:template match="db:link">
3884     <xsl:call-template name="compensate-space"/>
3885     <xsl:choose>
3886       <!-- with hyperref or hypdoc -->
3887       <xsl:when
3888         test="//db:seg[.='hyperref' or .= 'hypdoc']
3889           or
3890           $prepost/db:procedure[@xml:id='prepackage']
3891           /db:step[contains(@condition,'doc')
3892             and
3893             @remap='hypdoc']
3894           [db:constructorsynopsis='']">
3895         <xsl:text>\href{</xsl:text>
3896         <xsl:value-of select="@xlink:href"/>
3897         <xsl:text>}{</xsl:text>
3898       <xsl:choose>
3899         <xsl:when test="normalize-space(.)=''">
3900           <xsl:text>\nolinkurl{</xsl:text>
3901           <xsl:value-of
3902             select="replace(@xlink:href,'#','\#')"/>
3903           <xsl:text>}</xsl:text>
3904         </xsl:when>

```

```

3905         <xsl:otherwise>
3906             <xsl:apply-templates/>
3907         </xsl:otherwise>
3908     </xsl:choose>
3909     <xsl:text>}</xsl:text>
3910 </xsl:when>
3911 <xsl:otherwise>
3912     <xsl:choose>
3913         <!-- empty link text -->
3914         <xsl:when test=".=''">
3915             <xsl:text>\url{</xsl:text>
3916             <xsl:value-of select="@xlink:href"/>
3917             <xsl:text>}</xsl:text>
3918         </xsl:when>
3919         <xsl:otherwise>
3920             <xsl:text>{\color{blue}\uline{</xsl:text>
3921             <xsl:apply-templates/>
3922             <xsl:text>}}\footnote{\protect\url{</xsl:text>
3923             <xsl:value-of select="@xlink:href"/>
3924             <xsl:text>}}</xsl:text>
3925         </xsl:otherwise>
3926     </xsl:choose>
3927 </xsl:otherwise>
3928 </xsl:choose>
3929 <xsl:if
3930 test="contains('.,,:?!',
3931             substring(following-sibling::node()[1],1,1))">
3932     <xsl:text>\thinspace</xsl:text>
3933 </xsl:if>
3934 </xsl:template>

```

db:literallayout This element enables the inclusion of \LaTeX code that is to be passed through to the .dtx file untouched, for the sake of special effects or functions that cannot be modelled in *DocBook* XML. It also enables *bash* code to be passed to the output file as well as set in the documentation using the `\lstlisting` command (in restricted circumstances). Setting the `<audience>` attribute to `<none>` is unimplemented intended for use with a future HTML output where the \LaTeX code would be irrelevant.

```

3935 <xsl:template match="db:literallayout[not(@audience='none')]">
3936     <xsl:if test="parent::db:para">
3937         <xsl:call-template name="compensate-space"/>
3938     </xsl:if>
3939     <xsl:choose>
3940         <xsl:when test="@language='LaTeXe' or
3941             @language='LaTeX' or
3942             @language='TeX' or
3943             not(@language)">
3944             <xsl:value-of select="."/>
3945         </xsl:when>
3946         <xsl:when test="@language='bash'">
3947             <xsl:if
3948                 test="ancestor::db:part[@xml:id='code'] and

```

```

3949         ancestor::db:chapter/@xlink:href and
3950         ancestor::db:chapter/@xml:id and
3951         not(substring-after(
3952             ancestor::db:chapter/@xlink:href, '.')='dtd'
3953         or substring-after(
3954             ancestor::db:chapter/@xlink:href, '.')='xml'
3955         or substring-after(
3956             ancestor::db:chapter/@xlink:href, '.')='xsl'))">
3957         <xsl:text>% \iffalse&#xa;</xsl:text>
3958         <xsl:value-of select="$sentinel"/>
3959         <xsl:text>&#xa;</xsl:text>
3960         <xsl:apply-templates select="." mode="readme"/>
3961         <xsl:text>% \fi&#xa;</xsl:text>
3962     </xsl:if>
3963     <xsl:text>% \iffalse
3964 %&lt;*ignore>
3965 % \fi
3966 </xsl:text>
3967     <xsl:text>\begin{lstlisting}</xsl:text>
3968     <xsl:text>[basicstyle=\small\color{DarkGreen}</xsl:text>
3969     <xsl:text>\ttfamily,language=</xsl:text>
3970     <xsl:value-of select="/db:book/@userlevel"/>
3971     <xsl:text>]&#xa;</xsl:text>
3972     <xsl:value-of select="."/>
3973     <xsl:text>\end{lstlisting}&#xa;</xsl:text>
3974     <xsl:text>% \iffalse
3975 %&lt;/ignore>
3976 % \fi
3977 </xsl:text>
3978 </xsl:when>
3979 </xsl:choose>
3980 </xsl:template>

```

db:methodname Used to identify a template in documenting xSLT.

```

3981 <xsl:template match="db:methodname">
3982     <xsl:call-template name="compensate-space"/>
3983     <xsl:if test="not(@role) and not(ancestor::db:tgroup)">
3984         <xsl:text>\DescribeTemplate{</xsl:text>
3985         <xsl:value-of select="."/>
3986         <xsl:text>}</xsl:text>
3987     </xsl:if>
3988     <xsl:text>\texttt{</xsl:text>
3989     <xsl:value-of select="."/>
3990     <xsl:text>}</xsl:text>
3991 </xsl:template>

```

db:note [db:note with [not(parent::db:para)]]

Creates a sidebar box for a note.

```

3992 <xsl:template match="db:note[not(parent::db:para)]">
3993     <xsl:text>% \begin{Sbox}\begin{minipage}</xsl:text>
3994     <xsl:text>{.85\columnwidth}\raggedright\sffamily</xsl:text>

```

```

3995     <xsl:if test="@xml:id">
3996       <xsl:text>\label{</xsl:text>
3997       <xsl:value-of select="@xml:id"/>
3998       <xsl:text>}</xsl:text>
3999     </xsl:if>
4000     <xsl:text>&#xa;</xsl:text>
4001     <xsl:apply-templates/>
4002     <xsl:text>&#xa;% \end{minipage}\end{Sbox}</xsl:text>
4003     <xsl:text>{\fboxsep1em\centering\fbox{\TheSbox}</xsl:text>
4004     <xsl:text>\par\smallskip}&#xa;</xsl:text>
4005   </xsl:template>

```

db:note [db:note with [parent::db:para]]

Creates a sidebar box for a note.

```

4006   <xsl:template match="db:note[parent::db:para]">
4007     <xsl:choose>
4008       <xsl:when test="@role='marginal'">
4009         <xsl:text>\marginnote{\raggedright </xsl:text>
4010         <xsl:value-of select="normalize-space(.)"/>
4011         <xsl:text>}</xsl:text>
4012       </xsl:when>
4013       <xsl:otherwise>
4014         <xsl:call-template name="compensate-space"/>
4015         <xsl:text>{\bfseries </xsl:text>
4016         <xsl:apply-templates/>
4017         <xsl:text>}</xsl:text>
4018       </xsl:otherwise>
4019     </xsl:choose>
4020   </xsl:template>

```

db:option Identifies an optional command or value, usually to a L^AT_EX command if there is a @role attribute specified, and the element does not occur within a title, term, footnote, table, or figure. It is formatted separately using the exemplar commands \oarg and \marg where values are provided; otherwise output in bold monospace type.

```

4021   <xsl:template match="db:option">
4022     <xsl:call-template name="compensate-space"/>
4023     <xsl:if test="@role and
4024                 not(ancestor::db:title) and
4025                 not(ancestor::db:term) and
4026                 not(ancestor::db:footnote) and
4027                 not(ancestor::db:table) and
4028                 not(ancestor::db:informaltable) and
4029                 not(ancestor::db:figure)
4030                 and not(preceding-sibling::db:option
4031                        [.=current()/])">
4032       <xsl:text>\DescribeOption{</xsl:text>
4033       <xsl:value-of select="."/>
4034       <xsl:text>}</xsl:text>
4035     </xsl:if>

```

```

4036     <xsl:choose>
4037       <xsl:when test="@condition">
4038         <xsl:text>\oarg{</xsl:text>
4039         <xsl:value-of select="."/>
4040         <xsl:text>}</xsl:text>
4041       </xsl:when>
4042       <xsl:when test="@conformance">
4043         <xsl:text>\marg{</xsl:text>
4044         <xsl:value-of select="."/>
4045         <xsl:text>}</xsl:text>
4046       </xsl:when>
4047       <!-- otherwise it's for a package or command -->
4048       <xsl:otherwise>
4049         <xsl:text>\textbf{\texttt{</xsl:text>
4050         <xsl:if test="@language='bash'">
4051           <xsl:text>-</xsl:text>
4052         </xsl:if>
4053         <xsl:apply-templates/>
4054         <xsl:if test="@language='bash' and @conformance='colon'">
4055           <xsl:text>:</xsl:text>
4056         </xsl:if>
4057         <xsl:text>}}</xsl:text>
4058       </xsl:otherwise>
4059     </xsl:choose>
4060 </xsl:template>

```

db:package Identifies a L^AT_EX package and formats in sans-serif type. May also be used for other systems' packages, eg *bash*.

```

4061 <xsl:template match="db:package">
4062   <xsl:call-template name="compensate-space"/>
4063   <xsl:if test="@role and
4064               not(ancestor::db:title) and
4065               not(ancestor::db:abstract) and
4066               not(ancestor::db:acknowledgements) and
4067               not(ancestor::db:term) and
4068               not(ancestor::db:footnote) and
4069               not(ancestor::db:table) and
4070               not(ancestor::db:informaltable) and
4071               not(ancestor::db:figure)
4072               and not(preceding-sibling::db:package
4073                      [=current()/])">
4074     <xsl:text>\DescribePackage{</xsl:text>
4075     <xsl:choose>
4076       <xsl:when test=".=''">
4077         <xsl:value-of select="/*[1]/@xml:id"/>
4078       </xsl:when>
4079       <xsl:otherwise>
4080         <xsl:value-of select="."/>
4081       </xsl:otherwise>
4082     </xsl:choose>
4083     <xsl:text>}</xsl:text>
4084   </xsl:if>
4085   <!-- use sf but heavier when already sf -->

```

```

4086     <xsl:text>\textsf{</xsl:text>
4087     <xsl:choose>
4088       <xsl:when test=".= ' '">
4089         <xsl:value-of select="/*[1]/@xml:id"/>
4090       </xsl:when>
4091       <xsl:otherwise>
4092         <xsl:apply-templates/>
4093       </xsl:otherwise>
4094     </xsl:choose>
4095     <xsl:text>}</xsl:text>
4096   </xsl:template>

```

`db:personname` Identifies a person's name for indexing. The `<firstname>` and `<surname>` subelements **MUST** occur in that order, even if the individual's culture usually reverses them (eg Hungary).

```

4097   <xsl:template match="db:personname">
4098     <xsl:call-template name="compensate-space"/>
4099     <xsl:apply-templates/>
4100     <xsl:text>\index{</xsl:text>
4101     <xsl:value-of select="normalize-space(db:surname)"/>
4102     <xsl:text>!</xsl:text>
4103     <xsl:value-of select="normalize-space(db:firstname)"/>
4104     <xsl:if test="db:othername">
4105       <xsl:text> </xsl:text>
4106       <xsl:value-of select="normalize-space(db:othername)"/>
4107     </xsl:if>
4108     <xsl:text>}</xsl:text>
4109   </xsl:template>
4110
4111   <xsl:template match="db:personname/db:firstname">
4112     <xsl:value-of select="."/>
4113   </xsl:template>
4114
4115   <xsl:template match="db:personname/db:othername">
4116     <xsl:if test="preceding-sibling::node()!=' '">
4117       <xsl:text> </xsl:text>
4118     </xsl:if>
4119     <xsl:value-of select="."/>
4120     <xsl:if
4121       test="string-length(normalize-space()) = 1
4122           or
4123           (.=upper-case(.)
4124           and
4125           substring(.,string-length())!='.')">
4126       <xsl:text>.</xsl:text>
4127     </xsl:if>
4128   </xsl:template>
4129
4130   <xsl:template match="db:personname/db:surname">
4131     <xsl:if test="preceding-sibling::node()!=' '">
4132       <xsl:text> </xsl:text>
4133     </xsl:if>
4134     <xsl:value-of select="."/>

```

4135 </xsl:template>

db:primary Within an <indexterm>, identifies the word as the primary term.

```
4136     <xsl:template match="db:primary">
4137       <xsl:value-of select="normalize-space(.)"/>
4138       <xsl:if test="@sortas">
4139         <xsl:text>@</xsl:text>
4140         <xsl:value-of select="normalize-space(@sortas)"/>
4141       </xsl:if>
4142     </xsl:template>
```

db:productname Identifies a product name, either commercial or Open Source, especially program names. The @userlevel attribute can be used for the chapter of the Unix/GNU Linux manual.

```
4143     <xsl:template match="db:productname">
4144       <xsl:call-template name="compensate-space"/>
4145       <xsl:text>\emph{</xsl:text>
4146       <xsl:apply-templates/>
4147       <xsl:text>}</xsl:text>
4148       <xsl:if test="@userlevel">
4149         <xsl:text>\thinspace(</xsl:text>
4150         <xsl:value-of select="@userlevel"/>
4151         <xsl:text>)</xsl:text>
4152       </xsl:if>
4153     </xsl:template>
```

db:quotes [db:quote and db:phrase and db:wordasword]

Identifies quotes, words, or phrases which require quotation marks.

```
4154     <xsl:template match="db:quote | db:phrase | db:wordasword">
4155       <xsl:call-template name="compensate-space"/>
4156       <xsl:if test="count(preceding-sibling::node())=0">
4157         <xsl:text>\leavevmode\llap</xsl:text>
4158       </xsl:if>
4159       <xsl:text>`</xsl:text>
4160       <xsl:apply-templates/>
4161       <xsl:text>'</xsl:text>
4162       <xsl:if test="@linkend">
4163         <xsl:text>&#x0a;</xsl:text>
4164         <xsl:call-template name="makeref"/>
4165       </xsl:if>
4166     </xsl:template>
```

db:replaceable Like <token>, this represents variable information (usually mnemonic, like ‘your username’) for use in a <programlisting> element.

```
4167     <xsl:template match="db:replaceable">
4168       <xsl:call-template name="compensate-space"/>
4169       <!-- make sure we use the same delimiter
4170       as the parent element does -->
```

```

4171     <xsl:variable name="delim">
4172       <!-- in order of preference -->
4173       <xsl:choose>
4174         <xsl:when test="not(contains(.,'|'))">
4175           <xsl:text>|</xsl:text>
4176         </xsl:when>
4177         <xsl:when test="not(contains(.,'+'))">
4178           <xsl:text>+</xsl:text>
4179         </xsl:when>
4180         <xsl:otherwise>
4181           <xsl:text>`</xsl:text>
4182         </xsl:otherwise>
4183       </xsl:choose>
4184     </xsl:variable>
4185     <xsl:choose>
4186       <!-- stop-start \verb when in a command or systemitem -->
4187       <xsl:when test="parent::db:command
4188         or parent::db:systemitem
4189         or parent::db:filename">
4190         <xsl:value-of select="$delim"/>
4191         <xsl:text>{\ttfamily\itshape </xsl:text>
4192         <xsl:value-of select="normalize-space(.)"/>
4193         <xsl:text>}</xsl:text>
4194         <xsl:text>\verb</xsl:text>
4195         <xsl:value-of select="$delim"/>
4196       </xsl:when>
4197       <xsl:otherwise>
4198         <xsl:text>\uline{</xsl:text>
4199         <xsl:value-of select="normalize-space(.)"/>
4200         <xsl:text>}</xsl:text>
4201       </xsl:otherwise>
4202     </xsl:choose>
4203   </xsl:template>

```

secondary [db:secondary and db:tertiary]

Within an `<indexterm>`, identifies the word as the secondary (or tertiary) term for a multi-level index.

```

4204   <xsl:template match="db:secondary | db:tertiary">
4205     <xsl:text>!</xsl:text>
4206     <xsl:value-of select="normalize-space(.)"/>
4207     <xsl:if test="@sortas">
4208       <xsl:text>@</xsl:text>
4209       <xsl:value-of select="normalize-space(@sortas)"/>
4210     </xsl:if>
4211   </xsl:template>

```

db:see Within an `<indexterm>`, identifies the word to be the target of a ‘see’ link.

```

4212   <xsl:template match="db:see">
4213     <xsl:text>|see{</xsl:text>
4214     <xsl:value-of select="normalize-space(.)"/>
4215     <xsl:text>}</xsl:text>

```

4216 </xsl:template>

db:superscript **Identifies a superscript.**

```
4217     <xsl:template match="db:superscript">
4218         <xsl:text>\textsuperscript{</xsl:text>
4219         <xsl:apply-templates/>
4220         <xsl:text>}</xsl:text>
4221     </xsl:template>
```

db:markup **Identifies an element of markup in a non-XML language.**

```
4222     <xsl:template match="db:markup">
4223         <xsl:call-template name="compensate-space"/>
4224         <xsl:text>\texttt{</xsl:text>
4225         <xsl:apply-templates/>
4226         <xsl:text>}</xsl:text>
4227     </xsl:template>
```

db:tag **Identifies an item of XML construction. ONLY the name is given as content: the punctuation for elements, tags, attributes, etc, is added, based on the @class attribute. Different functions from DocTeX are used to format it, depending on this data, if the @role attribute is non-null.**

```
4228     <xsl:template match="db:tag">
4229         <xsl:call-template name="compensate-space"/>
4230         <xsl:variable name="tagval" select="normalize-space(.)"/>
4231         <!-- unclear what @role was destined for -->
4232         <xsl:if
4233             test="@role and
4234                 not(ancestor::db:title) and
4235                 not(ancestor::db:term) and
4236                 not(ancestor::db:footnote) and
4237                 not(ancestor::db:table) and
4238                 not(ancestor::db:informaltable) and
4239                 not(ancestor::db:figure)
4240                 and not(preceding-sibling::db:tag[.=$tagval])">
4241             <xsl:text>\Describe</xsl:text>
4242             <xsl:choose>
4243                 <xsl:when test="@class='element' or not(@class)">
4244                     <xsl:text>Element</xsl:text>
4245                 </xsl:when>
4246                 <xsl:when test="@class='attribute'">
4247                     <xsl:text>Attribute</xsl:text>
4248                 </xsl:when>
4249                 <xsl:when test="@class='attvalue'">
4250                     <xsl:text>AttributeValue</xsl:text>
4251                 </xsl:when>
4252                 <xsl:when test="@class='genentity'">
4253                     <xsl:text>Entity</xsl:text>
4254                 </xsl:when>
4255                 <xsl:otherwise>
4256                     <xsl:text>Error</xsl:text>
```

```

4257         </xsl:otherwise>
4258     </xsl:choose>
4259     <xsl:text>{</xsl:text>
4260     <xsl:apply-templates/>
4261     <xsl:text>}</xsl:text>
4262 </xsl:if>
4263 <xsl:text>\texttt{</xsl:text>
4264 <xsl:choose>
4265     <xsl:when test="@class='starttag'
4266             or
4267             @class='emptytag'
4268             or
4269             @class='endtag'
4270             or
4271             not(@class)'">
4272     <xsl:text>&lt;</xsl:text>
4273     <xsl:if test="@class='endtag'">
4274         <xsl:text></xsl:text>
4275     </xsl:if>
4276 </xsl:when>
4277 <xsl:when test="@class='attvalue'">
4278     <xsl:if
4279         test="name(preceding-sibling::node()[1])='tag'
4280             and preceding-sibling::*[1]/@class='attribute'">
4281         <xsl:text>\unskip</xsl:text>
4282     </xsl:if>
4283     <xsl:text>"</xsl:text>
4284 </xsl:when>
4285 <xsl:when
4286     test="@class='attribute' and not(@conformance='noat')">
4287     <xsl:if
4288         test="name(preceding-sibling::node()[1])='tag'
4289             and (
4290                 preceding-sibling::*[1]/@class='element' or
4291                 preceding-sibling::*[1]/@class='emptytag' or
4292                 preceding-sibling::*[1]/@class='starttag' or
4293                 preceding-sibling::*[1][not(@class)]
4294             )">
4295         <xsl:text>\unskip</xsl:text>
4296     </xsl:if>
4297     <xsl:text>@</xsl:text>
4298 </xsl:when>
4299 <xsl:when test="@class='genentity'">
4300     <xsl:text>\&#x</xsl:text>
4301 </xsl:when>
4302 <xsl:when test="@class='declaration'">
4303     <xsl:text>!</xsl:text>
4304 </xsl:when>
4305 <xsl:when test="@class='numcharref'">
4306     <xsl:text>\&#x</xsl:text>
4307 </xsl:when>
4308 </xsl:choose>
4309 <xsl:value-of select="$tagval"/>
4310 </xsl:choose>

```

```

4311     <xsl:when test="@class='starttag'
4312                 or
4313                 @class='emptytag'
4314                 or
4315                 @class='endtag'
4316                 or
4317                 not(@class)">
4318         <xsl:if test="@class='emptytag'">
4319             <xsl:text>/</xsl:text>
4320         </xsl:if>
4321         <xsl:text>&gt;</xsl:text>
4322     </xsl:when>
4323     <xsl:when test="@class='attvalue'">
4324         <xsl:text>"</xsl:text>
4325     </xsl:when>
4326     <xsl:when test="@class='genentity' or @class='numcharref'">
4327         <xsl:text>;</xsl:text>
4328     </xsl:when>
4329 </xsl:choose>
4330 <xsl:text>}</xsl:text>
4331 <!-- INDEXING MARKUP -->
4332 <xsl:text>\index{</xsl:text>
4333 <!-- sort value -->
4334 <xsl:value-of select="$tagval"/>
4335 <!-- style for entry (different delimiter to makeindex default) -->
4336 <xsl:text>=\texttt{</xsl:text>
4337 <!-- STAGO -->
4338 <xsl:choose>
4339     <xsl:when test="@class='element' or not(@class)">
4340         <xsl:text>\char'074 </xsl:text>
4341     </xsl:when>
4342     <!-- safety first in outputting an @ sign in an index -->
4343     <xsl:when test="@class='attribute'">
4344         <xsl:text>\char'100 </xsl:text>
4345     </xsl:when>
4346     <xsl:when test="@class='attvalue'">
4347         <xsl:text>"</xsl:text>
4348     </xsl:when>
4349     <xsl:when test="@class='emptytag'">
4350         <xsl:text>\char'074 </xsl:text>
4351     </xsl:when>
4352     <xsl:when test="@class='genentity'">
4353         <xsl:text>\&lt;</xsl:text>
4354     </xsl:when>
4355     <xsl:when test="@class='localname'">
4356         <xsl:text></xsl:text>
4357     </xsl:when>
4358     <xsl:when test="@class='starttag'">
4359         <xsl:text>\char'074 </xsl:text>
4360     </xsl:when>
4361     <xsl:otherwise>
4362         <xsl:text>?</xsl:text>
4363     </xsl:otherwise>
4364 </xsl:choose>

```

```

4365     <xsl:value-of select="$tagval"/>
4366     <!-- TAGC -->
4367     <xsl:choose>
4368         <xsl:when test="@class='element' or not(@class)">
4369             <xsl:text>\char'Ø76 </xsl:text>
4370         </xsl:when>
4371         <xsl:when test="@class='attribute'">
4372             <xsl:text></xsl:text>
4373         </xsl:when>
4374         <xsl:when test="@class='attvalue'">
4375             <xsl:text>"</xsl:text>
4376         </xsl:when>
4377         <xsl:when test="@class='emptytag'">
4378             <xsl:text>\char'Ø76 </xsl:text>
4379         </xsl:when>
4380         <xsl:when test="@class='genentity'">
4381             <xsl:text>;</xsl:text>
4382         </xsl:when>
4383         <xsl:when test="@class='localname'">
4384             <xsl:text></xsl:text>
4385         </xsl:when>
4386         <xsl:when test="@class='starttag'">
4387             <xsl:text>\char'Ø76 </xsl:text>
4388         </xsl:when>
4389         <xsl:otherwise>
4390             <xsl:text>?</xsl:text>
4391         </xsl:otherwise>
4392     </xsl:choose>
4393     <xsl:text>} (</xsl:text>
4394     <!-- human names -->
4395     <xsl:choose>
4396         <xsl:when test="@class='element' or not(@class)">
4397             <xsl:text>element</xsl:text>
4398         </xsl:when>
4399         <xsl:when test="@class='attribute'">
4400             <xsl:text>attribute</xsl:text>
4401         </xsl:when>
4402         <xsl:when test="@class='attvalue'">
4403             <xsl:text>attribute value</xsl:text>
4404         </xsl:when>
4405         <xsl:when test="@class='emptytag'">
4406             <xsl:text>empty element</xsl:text>
4407         </xsl:when>
4408         <xsl:when test="@class='genentity'">
4409             <xsl:text>general entity</xsl:text>
4410         </xsl:when>
4411         <xsl:when test="@class='localname'">
4412             <xsl:text>local name</xsl:text>
4413         </xsl:when>
4414         <xsl:when test="@class='starttag'">
4415             <xsl:text>start-tag</xsl:text>
4416         </xsl:when>
4417         <xsl:otherwise>
4418             <xsl:value-of select="@class"/>

```

```

4419     </xsl:otherwise>
4420 </xsl:choose>
4421 <!-- two-level index -->
4422 <xsl:text>}}\index{</xsl:text>
4423 <!-- top level: pluralised class with colon -->
4424 <xsl:choose>
4425   <xsl:when test="@class='element' or not(@class)">
4426     <xsl:text>elements</xsl:text>
4427   </xsl:when>
4428   <xsl:when test="@class='attribute'">
4429     <xsl:text>attributes</xsl:text>
4430   </xsl:when>
4431   <xsl:when test="@class='attvalue'">
4432     <xsl:text>attribute values</xsl:text>
4433   </xsl:when>
4434   <xsl:when test="@class='emptytag'">
4435     <xsl:text>empty elements</xsl:text>
4436   </xsl:when>
4437   <xsl:when test="@class='genentity'">
4438     <xsl:text>general entities</xsl:text>
4439   </xsl:when>
4440   <xsl:when test="@class='localname'">
4441     <xsl:text>local names</xsl:text>
4442   </xsl:when>
4443   <xsl:when test="@class='starttag'">
4444     <xsl:text>start-tags</xsl:text>
4445   </xsl:when>
4446   <xsl:otherwise>
4447     <xsl:value-of select="@class"/>
4448     <xsl:text>s</xsl:text>
4449   </xsl:otherwise>
4450 </xsl:choose>
4451 <!-- colon and second-level delimiter -->
4452 <xsl:text>:!</xsl:text>
4453 <!-- sort value -->
4454 <xsl:value-of select="$tagval"/>
4455 <xsl:text>=\texttt{</xsl:text>
4456 <!-- STAG0 -->
4457 <xsl:choose>
4458   <xsl:when test="@class='element' or not(@class)">
4459     <xsl:text>\char'074 </xsl:text>
4460   </xsl:when>
4461   <xsl:when test="@class='attribute'">
4462     <xsl:text>\char'100 </xsl:text>
4463   </xsl:when>
4464   <xsl:when test="@class='attvalue'">
4465     <xsl:text>"</xsl:text>
4466   </xsl:when>
4467   <xsl:when test="@class='emptytag'">
4468     <xsl:text>\char'074 </xsl:text>
4469   </xsl:when>
4470   <xsl:when test="@class='genentity'">
4471     <xsl:text>\&lt;</xsl:text>
4472   </xsl:when>

```

```

4473     <xsl:when test="@class='localname'">
4474         <xsl:text></xsl:text>
4475     </xsl:when>
4476     <xsl:when test="@class='starttag'">
4477         <xsl:text>\char'Ø74 </xsl:text>
4478     </xsl:when>
4479     <xsl:otherwise>
4480         <xsl:text>?</xsl:text>
4481     </xsl:otherwise>
4482 </xsl:choose>
4483 <xsl:value-of select="$tagval"/>
4484 <!-- TAGC -->
4485 <xsl:choose>
4486     <xsl:when test="@class='element' or not(@class)">
4487         <xsl:text>\char'Ø76 </xsl:text>
4488     </xsl:when>
4489     <xsl:when test="@class='attribute'">
4490         <xsl:text></xsl:text>
4491     </xsl:when>
4492     <xsl:when test="@class='attvalue'">
4493         <xsl:text>"</xsl:text>
4494     </xsl:when>
4495     <xsl:when test="@class='emptytag'">
4496         <xsl:text>/\char'Ø76</xsl:text>
4497     </xsl:when>
4498     <xsl:when test="@class='genentity'">
4499         <xsl:text>;</xsl:text>
4500     </xsl:when>
4501     <xsl:when test="@class='localname'">
4502         <xsl:text></xsl:text>
4503     </xsl:when>
4504     <xsl:when test="@class='starttag'">
4505         <xsl:text>\char'Ø76</xsl:text>
4506     </xsl:when>
4507     <xsl:otherwise>
4508         <xsl:text>?</xsl:text>
4509     </xsl:otherwise>
4510 </xsl:choose>
4511 <xsl:text>}}</xsl:text>
4512 <xsl:if
4513     test="name(following-sibling::node())[1]='' and
4514         starts-with(following-sibling::node())[1],'s')">
4515     <xsl:text>\kern1pt{}</xsl:text>
4516 </xsl:if>
4517 </xsl:template>

```

db:task Highlights TODO notes like

db:task **TODO: Collate these into a plan**

```

4518     <xsl:template match="db:task">
4519         <xsl:text>% \par\smallskip\leavevmode
4520     % \CPKrunningecho{</xsl:text>

```

```

4521     <xsl:value-of
4522         select="parent::db:annotation/@xreflabel
4523             |
4524             ancestor::db:varlistentry/db:term"/>
4525     <xsl:text>}%
4526 % \leavevmode\kern-12pt\colorbox{Black}{\vbox{\advance\hsize by-2em
4527 % \color{White}\sffamily\raggedright TOD0:</xsl:text>
4528     <xsl:for-each select="*">
4529         <xsl:text> </xsl:text>
4530         <xsl:value-of select="normalize-space(.)"/>
4531     </xsl:for-each>
4532     <xsl:text>}}\par\smallskip&#xa;</xsl:text>
4533 </xsl:template>

```

db:title [in db:note/]

Makes the title of a note a subsubsection level heading.

```

4534 <xsl:template match="db:note/db:title">
4535     <xsl:text>% \subsubsection*{</xsl:text>
4536     <xsl:apply-templates/>
4537     <xsl:text>}&#xa;</xsl:text>
4538 </xsl:template>

```

db:token Identifies atomic values within the text of a <programlisting> element and highlights them in italics.

```

4539 <xsl:template match="db:token">
4540     <xsl:call-template name="compensate-space"/>
4541     <xsl:text>{\normalfont\itshape</xsl:text>
4542     <xsl:apply-templates/>
4543     <xsl:text>}</xsl:text>
4544 </xsl:template>

```

db:type [with [@role='font' or not(@role)]]

Usurped to represent typographic variation when documenting typesetting. See the doctexbook.dtd for details of the attributes added.

```

4545 <xsl:template match="db:type[@role='font' or not(@role)]">
4546     <xsl:text>{</xsl:text>
4547     <xsl:choose>
4548         <!-- empty and fontfamily means identify the font -->
4549         <xsl:when test=".=' ' and @fontfamily">
4550             <xsl:choose>
4551                 <xsl:when test="@fontfamily='sans'">
4552                     <xsl:text>\sffamily </xsl:text>
4553                 </xsl:when>
4554                 <xsl:when test="@fontfamily='roman'">
4555                     <xsl:text>\rmfamily </xsl:text>
4556                 </xsl:when>
4557                 <xsl:when test="@fontfamily='monospace'">
4558                     <xsl:text>\ttfamily </xsl:text>

```

```

4559         </xsl:when>
4560     </xsl:choose>
4561     <xsl:text>\printexternalcurrentfont</xsl:text>
4562 </xsl:when>
4563 <!-- otherwise format something -->
4564 <xsl:otherwise>
4565     <xsl:if test="@fontcolor">
4566         <xsl:text>\color</xsl:text>
4567         <xsl:if test="@conformance">
4568             <xsl:text>[</xsl:text>
4569             <xsl:value-of select="@conformance"/>
4570             <xsl:text>]</xsl:text>
4571         </xsl:if>
4572         <xsl:text>{</xsl:text>
4573         <xsl:value-of select="@fontcolor"/>
4574         <xsl:text>}</xsl:text>
4575     </xsl:if>
4576     <xsl:if test="@fontencoding">
4577         <xsl:text>\fontencoding{</xsl:text>
4578         <xsl:value-of select="upper-case(@fontencoding)"/>
4579         <xsl:text>}</xsl:text>
4580     </xsl:if>
4581     <xsl:if test="@fontfamily">
4582         <!-- special cases -->
4583         <xsl:choose>
4584             <xsl:when test="@fontfamily='sans'">
4585                 <xsl:text>\sffamily </xsl:text>
4586             </xsl:when>
4587             <xsl:when test="@fontfamily='roman'">
4588                 <xsl:text>\rmfamily </xsl:text>
4589             </xsl:when>
4590             <xsl:when test="@fontfamily='monospace'">
4591                 <xsl:text>\ttfamily </xsl:text>
4592             </xsl:when>
4593         </xsl:choose>
4594     </xsl:if>
4595     <xsl:if test="@fontface">
4596         <xsl:text>\fontfamily{</xsl:text>
4597         <xsl:value-of select="@fontface"/>
4598         <xsl:text>}</xsl:text>
4599     </xsl:if>
4600     <xsl:if test="@fontshape">
4601         <xsl:text>\</xsl:text>
4602         <xsl:value-of select="@fontshape"/>
4603         <xsl:text>shape</xsl:text>
4604     </xsl:if>
4605     <xsl:if test="@fontseries">
4606         <xsl:text>\fontseries{</xsl:text>
4607         <xsl:value-of select="@fontseries"/>
4608         <xsl:text>}</xsl:text>
4609     </xsl:if>
4610     <xsl:if test="@fontsize">
4611         <xsl:text>\fontsize{</xsl:text>
4612         <xsl:choose>

```

```

4613         <xsl:when test="contains(@fontsize, '/')">
4614             <xsl:value-of
4615                 select="substring-before(@fontsize, '/')"/>
4616         </xsl:when>
4617         <xsl:otherwise>
4618             <xsl:value-of select="@fontsize"/>
4619         </xsl:otherwise>
4620     </xsl:choose>
4621     <xsl:text>}{</xsl:text>
4622     <xsl:choose>
4623         <xsl:when test="contains(@fontsize, '/')">
4624             <xsl:value-of
4625                 select="substring-after(@fontsize, '/')"/>
4626         </xsl:when>
4627         <xsl:otherwise>
4628             <xsl:text>Ø</xsl:text>
4629         </xsl:otherwise>
4630     </xsl:choose>
4631     <xsl:text>}</xsl:text>
4632 </xsl:if>
4633 <xsl:text>\selectfont </xsl:text>
4634 <xsl:apply-templates/>
4635 <xsl:if test="@fontsize and contains(@fontsize, '/')">
4636     <xsl:text>\par</xsl:text>
4637 </xsl:if>
4638 </xsl:otherwise>
4639 </xsl:choose>
4640 <xsl:text>}</xsl:text>
4641 </xsl:template>

```

db:typecol [**db:type** with [**@role='color'**]]

Used when the <type> element invokes colour alone, and generates the requested colour for use with the xcolor package

```

4642 <xsl:template match="db:type[@role='color']">
4643     <xsl:call-template name="compensate-space"/>
4644     <xsl:text>\DescribeColor{</xsl:text>
4645     <xsl:value-of select="."/>
4646     <xsl:text>}\texttt{</xsl:text>
4647     <xsl:value-of select="."/>
4648     <xsl:text>}</xsl:text>
4649 </xsl:template>

```

db:uri with an @xlink:href URI *and* text content, this formats the content followed by a footnote to the URI. Without the attribute, the content itself is assumed by be a URI and is formatted inline. An optional @xreflabel attribute holding a Usenet News group name, with the @type attribute set to "news", will add a reference to the group.

```

4650 <xsl:template match="db:uri">
4651     <xsl:call-template name="compensate-space"/>
4652     <xsl:choose>

```

```

4653     <xsl:when test="@xlink:href and .!=''">
4654         <xsl:apply-templates/>
4655         <xsl:text>\footnote{\url{</xsl:text>
4656         <xsl:value-of select="@xlink:href"/>
4657         <xsl:text>}}</xsl:text>
4658     </xsl:when>
4659     <xsl:when test="not(@xlink:href) and .!=''">
4660         <xsl:text>\url{</xsl:text>
4661         <xsl:if test="@type">
4662             <xsl:value-of select="@type"/>
4663             <xsl:text>:</xsl:text>
4664         </xsl:if>
4665         <xsl:apply-templates/>
4666         <xsl:text>}</xsl:text>
4667     </xsl:when>
4668 </xsl:choose>
4669 <xsl:if test="@xreflabel and @type='news'">
4670     <xsl:text> at \texttriangle\verb`</xsl:text>
4671     <xsl:value-of select="@xreflabel"/>
4672     <xsl:text>\texttriangle{}</xsl:text>
4673 </xsl:if>
4674 </xsl:template>

```

`db:varname` [`db:varname` with [`ancestor::db:footnote` or `ancestor::db:abstract`] and `db:parameter` with [`ancestor::db:footnote` or `ancestor::db:abstract`]

Identifies variables and parameters, especially for \LaTeX with the `@role` set to "counter", "length", "template"; or "variable" or `<parameter>` (for XSLT), using the relevant `\Describe` macro to put the unheralded name in the margin. For other values (color and prog), this invokes the `avoidverb` named template for formatting.

A feature is the avoidance of clashes with a containing `<annotation>` for the same role and name: the `\Describe` macro is invoked only where the `@role` attribute is non-null

and the `@condition` is *not* set to "nodesc"

and it is not in the first paragraph of the `<annotation>`

and the `@role` (or, in the case of the `<parameter>`, the name) is not the same value of the `@role` attribute on the `<annotation>`

and the content is not the same as the `<annotation>`'s `@xreflabel` attribute.

```

4675 <!-- Counters and Lengths and Variables -->
4676 <xsl:template
4677     match="db:varname[not(ancestor::db:footnote
4678                             or ancestor::db:abstract)]
4679         |
4680         db:parameter[not(ancestor::db:footnote
4681                             or ancestor::db:abstract)]">
4682     <xsl:call-template name="compensate-space"/>
4683     <xsl:if
4684         test="@role and @role!=''
4685             and
4686             not(@condition='nodesc')

```

```

4687         and
4688         not(count(parent::db:para
4689                 /preceding-sibling::db:para) = 0
4690         and
4691         ancestor::db:annotation
4692         [(@role=current()/@role
4693         or
4694         @role=current()/name())
4695         and
4696         @xreflabel=current()/.)])
4697         and
4698         not(preceding-sibling::db:varname
4699         [.=current()/.]
4700         [@role=current()/@role]))">
4701     <xsl:text>\Describe</xsl:text>
4702     <!-- Counter Function Length Template Variable Parameter -->
4703     <xsl:value-of
4704         select="translate(
4705             substring(@role,1,1),'cfltp','CFLTP')"/>
4706     <xsl:value-of select="substring(@role,2)"/>
4707     <xsl:text>{</xsl:text>
4708     <xsl:value-of select="normalize-space(.)"/>
4709     <xsl:text>}</xsl:text>
4710 </xsl:if>
4711 <xsl:call-template name="avoidverb"/>
4712 </xsl:template>
4713
4714 <xsl:template
4715     match="db:varname[ancestor::db:footnote
4716                     or
4717                     ancestor::db:abstract]
4718         |
4719         db:parameter[ancestor::db:footnote
4720                     or
4721                     ancestor::db:abstract]"/>
4722     <xsl:call-template name="compensate-space"/>
4723     <xsl:call-template name="avoidverb"/>
4724 </xsl:template>

```

corefs Counted references (corefs) are for assigning a number to items in an unnumbered list so that they can be referenced with an ordinal.

```

4725 <xsl:template match="db:coref">
4726     <xsl:text>\setcounter{CPKcoref}{</xsl:text>
4727     <xsl:value-of
4728         select="count(//*[ @xml:id=current()/@linkend]
4729                 /descendant::*[name()=current()/@xreflabel])"/>
4730     <xsl:text>}\numberstring{CPKcoref}</xsl:text>
4731 </xsl:template>

```

revision List the revisions as unnumbered subsections, showing the version and date; if there's only one linked change, use a bullet; otherwise use numbers.

```

4732 <xsl:template match="db:revision">
4733   <xsl:text>% \subsection*{v.</xsl:text>
4734   <xsl:value-of select="@version"/>
4735   <xsl:text> (</xsl:text>
4736   <xsl:value-of select="db:date/@YYYY-MM-DD"/>
4737   <xsl:text>))&#xa;</xsl:text>
4738   <xsl:apply-templates select="db:revdescription/*"/>
4739   <xsl:if test="//*[@revisionflag and
4740     @revision=current()/db:date/@YYYY-MM-DD]">
4741     <xsl:text>% \begin{</xsl:text>
4742     <xsl:choose>
4743       <xsl:when
4744         test="count(//*[@revisionflag and
4745           @revision=current()/db:date/@YYYY-MM-DD])>1">
4746         <xsl:text>enumerate</xsl:text>
4747       </xsl:when>
4748       <xsl:otherwise>
4749         <xsl:text>itemize</xsl:text>
4750       </xsl:otherwise>
4751     </xsl:choose>
4752     <xsl:text>}[nosep]&#xa;</xsl:text>
4753     <xsl:for-each
4754       select="//*[@revisionflag and
4755         @revision=current()/db:date/@YYYY-MM-DD]">
4756       <xsl:text>% \item \vref{</xsl:text>
4757       <xsl:value-of select="@xml:id"/>
4758       <xsl:text>}</xsl:text>
4759       <xsl:value-of select="@revisionflag"/>
4760       <xsl:text>: </xsl:text>
4761       <xsl:value-of select="normalize-space(@annotations)"/>
4762       <xsl:text>&#xa;</xsl:text>
4763     </xsl:for-each>
4764     <xsl:text>% \end{</xsl:text>
4765     <xsl:choose>
4766       <xsl:when
4767         test="count(//*[@revisionflag
4768           and
4769           @revision=current()
4770           /db:date/@YYYY-MM-DD])>1">
4771       <xsl:text>enumerate</xsl:text>
4772     </xsl:when>
4773     <xsl:otherwise>
4774       <xsl:text>itemize</xsl:text>
4775     </xsl:otherwise>
4776   </xsl:choose>
4777   <xsl:text>}&#xa;</xsl:text>
4778 </xsl:if>
4779 </xsl:template>

```

`inlineequation` Inline equations MUST contain \LaTeX math code, so they are rendered within inline math delimiters.

```

4780 <xsl:template match="db:inlineequation">
4781   <xsl:apply-templates/>

```

```

4782     <xsl:if test="@xml:id">
4783       <xsl:text>\label{</xsl:text>
4784       <xsl:value-of select="@xml:id"/>
4785       <xsl:text>}</xsl:text>
4786     </xsl:if>
4787   </xsl:template>
4788
4789   <xsl:template match="db:mathphrase">
4790     <xsl:choose>
4791       <!-- assume @remap is LaTeX -->
4792       <xsl:when test="@remap">
4793         <xsl:text>\(</xsl:text>
4794         <xsl:value-of select="@remap"/>
4795         <xsl:text>\)</xsl:text>
4796       </xsl:when>
4797       <xsl:otherwise>
4798         <xsl:apply-templates/>
4799       </xsl:otherwise>
4800     </xsl:choose>
4801   </xsl:template>

```

db:xref Cross-references are done in the standard ID/IDREF manner, except that the @linkend attribute on the <xref> element type has been redeclared as IDREFS to allow references to multiple targets, although this is not yet fully implemented.

```

4802   <xsl:template match="db:xref[@linkend]">
4803     <xsl:call-template name="compensate-space"/>
4804     <!-- remember where we started -->
4805     <xsl:variable name="context" select="."/>
4806     <!-- visit each target in turn -->
4807     <xsl:for-each select="tokenize(normalize-space(@linkend),' ')">
4808       <xsl:variable name="idref" select="."/>
4809       <xsl:variable name="target"
4810         select="$thisdoc//*[&#x0000;@xml:id=$idref]"/>
4811       <xsl:if test="position()>1">
4812         <xsl:if test="position()!<#x0000;last()<#x0000;">
4813           <xsl:text>; </xsl:text>
4814         </xsl:if>
4815         <xsl:if test="position()<#x0000;last()<#x0000;">
4816           <xsl:text> and </xsl:text>
4817         </xsl:if>
4818       </xsl:if>

```

The targets which the varioref package does not yet handle come first in the choose/when, and the remainder afterwards. Catch items in a description list before anything else, and give them an ordinal number.

```

4819     <!-- for normal references -->
4820     <xsl:if test="not($context/@role='short')">
4821       <xsl:choose>
4822         <!-- WHOLE DESCRIPTION LIST -->
4823         <xsl:when test="name($target)='variablelist'">
4824           <xsl:text>the list in </xsl:text>
4825         </xsl:when>

```

```

4826      <!-- DESCRIPTION LIST ENTRY-->
4827      <xsl:when test="name($target)='varlistentry'">
4828        <xsl:text>\textbf{\textsf{</xsl:text>
4829        <xsl:apply-templates
4830          select="$target/db:term/node()"/>
4831        <xsl:text>}}'</xsl:text>
4832      <!-- add ordinal if NOT a pageref -->
4833      <xsl:if test="not($context/@xrefstyle='page')">
4834        <xsl:text>, the </xsl:text>
4835        <xsl:variable name="nth"
4836          select="number(count($target/
4837            preceding-sibling::db:varlistentry)+1)"/>
4838        <xsl:choose>
4839          <!-- 1st -->
4840          <xsl:when test="$nth = 1">
4841            <xsl:text>first</xsl:text>
4842          </xsl:when>
4843          <!-- last -->
4844          <xsl:when
4845            test="$nth = count($target/
4846              parent::db:variablelist/db:varlistentry)">
4847            <xsl:text>last</xsl:text>
4848          </xsl:when>
4849          <!-- digits over 12 -->
4850          <xsl:when test="$nth > 12">
4851            <xsl:number value="$nth"
4852              ordinal="yes" format="1"/>
4853          </xsl:when>
4854          <!-- words under/= 12 -->
4855          <xsl:otherwise>
4856            <xsl:number value="$nth"
4857              ordinal="yes" format="w"/>
4858          </xsl:otherwise>
4859        </xsl:choose>
4860        <xsl:text> item in </xsl:text>
4861        <xsl:choose>
4862          <!-- same list -->
4863          <xsl:when
4864            test="generate-id($context
4865              /ancestor::db:variablelist[1])
4866              =
4867              generate-id($target
4868                /parent::db:variablelist)">
4869            <xsl:text>this list</xsl:text>
4870          </xsl:when>
4871          <!-- different list -->
4872          <xsl:otherwise>
4873            <xsl:text>the list in </xsl:text>
4874          </xsl:otherwise>
4875        </xsl:choose>
4876      </xsl:if>
4877      <!-- add afterwards, on p.nn or in sec ss-->
4878    </xsl:when><!-- END DESC LIST ENTRY -->

```

Unnumbered list items also need an ordinal reference.

```

4879      <!-- UNNUMBERED LIST ITEMS NEED HELP -->
4880      <xsl:when
4881        test="name($target)='listitem' and
4882              name($target/ancestor::*[1])='itemizedlist'">
4883        <xsl:text>the </xsl:text>
4884        <xsl:variable name="nth"
4885          select="number(count($target/
4886            preceding-sibling::db:listitem)+1)"/>
4887        <xsl:choose>
4888          <xsl:when test="$nth=1">
4889            <xsl:text>first</xsl:text>
4890          </xsl:when>
4891          <xsl:when
4892            test="$nth=count($target/
4893              ancestor::db:itemizedlist[1]/db:listitem)">
4894            <xsl:text>last</xsl:text>
4895          </xsl:when>
4896          <!-- from the fmtcount package -->
4897          <xsl:otherwise>
4898            <xsl:text>\ordinal</xsl:text>
4899            <xsl:if test="$nth < 10">
4900              <xsl:text>string</xsl:text>
4901            </xsl:if>
4902            <xsl:text>num{</xsl:text>
4903            <xsl:value-of select="$nth"/>
4904            <xsl:text>}</xsl:text>
4905          </xsl:otherwise>
4906        </xsl:choose>
4907        <xsl:text> item in the list </xsl:text>
4908        <xsl:if test="not($context/@xrefstyle='page')">
4909          <xsl:text>in </xsl:text>
4910        </xsl:if>
4911      </xsl:when>

```

Ordered lists and procedures already have numbers.

```

4912      <!-- ORDERED LIST ITEMS or PROCEDURE STEPS -->
4913      <xsl:when
4914        test="(local-name($target)='listitem' and
4915              local-name($target/parent::*[1])='orderedlist')
4916              or
4917              (local-name($target)='step' and
4918              local-name($target/parent::*[1])='procedure')">
4919        <xsl:choose>
4920          <xsl:when test="$target/ancestor::procedure">
4921            <xsl:text>step</xsl:text>
4922          </xsl:when>
4923          <xsl:otherwise>
4924            <xsl:text>item</xsl:text>
4925          </xsl:otherwise>
4926        </xsl:choose>
4927        <xsl:text>~\ref{</xsl:text>
4928        <xsl:value-of select="$idref"/>

```

```

4929      <xsl:text>></xsl:text>
4930      <xsl:choose>
4931        <!-- same list -->
4932        <xsl:when
4933          test="($target/ancestor::db:procedure and
4934            generate-id($context
4935              /ancestor::db:procedure[1])=
4936            generate-id($target
4937              /parent::db:procedure)
4938          )
4939          or
4940          ($context/ancestor::db:orderedlist and
4941            generate-id($context
4942              /ancestor::db:orderedlist[1])=
4943            generate-id($target
4944              /parent::db:orderedlist))">
4945          <xsl:choose>
4946            <xsl:when
4947              test="count($context/preceding::*
4948                |
4949                $target)=count($context/preceding:*)">
4950              <xsl:text> above</xsl:text>
4951            </xsl:when>
4952            <xsl:otherwise>
4953              <xsl:text> below</xsl:text>
4954            </xsl:otherwise>
4955          </xsl:choose>
4956        </xsl:when>
4957        <!-- elsewhere -->
4958        <xsl:otherwise>
4959          <xsl:text> in the list on </xsl:text>
4960          <xsl:text>p.\thinspace\pageref{</xsl:text>
4961          <xsl:value-of select="$idref"/>
4962          <xsl:text>}</xsl:text>
4963        </xsl:otherwise>
4964      </xsl:choose>
4965    </xsl:when>

```

References to *whole* ordered lists, informal tables, code listings, and notes, warnings, and sidebars just need an identity.

```

4966      <!-- TITLED LISTS -->
4967      <xsl:when
4968        test="(local-name($target)='orderedlist'
4969          or
4970          local-name($target)='itemizedlist')
4971          and $target/db:title">
4972        <xsl:text>the list `</xsl:text>
4973        <xsl:apply-templates
4974          select="$target/db:title/node()"/>
4975        <xsl:text>' in </xsl:text>
4976      </xsl:when>
4977      <!-- UNTITLED LISTS -->
4978      <xsl:when

```

```

4979         test="local-name($target)='orderedlist'
4980         or
4981         local-name($target)='itemizedlist'">
4982         <xsl:text>the list in </xsl:text>
4983     </xsl:when>
4984     <!-- INFORMAL TABLES -->
4985     <xsl:when
4986         test="local-name($target)='informaltable'">
4987         <xsl:text>the table </xsl:text>
4988     </xsl:when>
4989     <!-- PROGRAMLISTING/@condition="ignore"
4990         in package code -->
4991     <xsl:when
4992         test="local-name($target)='programlisting'">
4993         <xsl:text>the code </xsl:text>
4994     </xsl:when>
4995     <!-- NOTE, WARNING, SIDEBAR -->
4996     <xsl:when
4997         test="local-name($target)='note' or
4998             local-name($target)='warning' or
4999             local-name($target)='sidebar'">
5000         <xsl:text>the </xsl:text>
5001         <xsl:value-of select="local-name($target)"/>
5002         <xsl:text> in </xsl:text>
5003     </xsl:when>
5004     <!-- there is no otherwise -->
5005 </xsl:choose>
5006 </xsl:if>

```

References to extractable files, and page references; but omit the automated reference for the list items already done.

```

5007     <!-- AUTOMATED or special case of ref to extractable file -->
5008     <xsl:choose>
5009         <xsl:when
5010             test="$target/ancestor::db:part[@xml:id='files']
5011                 and
5012                 $target/@xlink:href">
5013             <xsl:text>\url{</xsl:text>
5014             <xsl:value-of select="$target/@xlink:href"/>
5015             <xsl:text>}</xsl:text>
5016         </xsl:when>
5017         <xsl:when
5018             test="$context/@xrefstyle='page'
5019                 or local-name($target)='informaltable'
5020                 or local-name($target)='programlisting'">
5021             <xsl:text>\vpageref</xsl:text>
5022         </xsl:when>
5023         <!-- omit numbered list items done manually above -->
5024         <xsl:when
5025             test="(local-name($target)='listitem' and
5026                 local-name($target/ancestor::*[1])='orderedlist')
5027                 or
5028                 ($context/ancestor::db:procedure and

```

```

5029         generate-id($context/ancestor::db:procedure)=
5030         generate-id($target/parent::db:procedure))">
5031     <xsl:text></xsl:text>
5032 </xsl:when>
5033 <!-- exclude link when varlistentries are
5034       in the same variablelist (so no \vref) -->
5035 <xsl:when
5036     test="$context/ancestor::db:variablelist
5037           and
5038           name($target)='varlistentry'
5039           and
5040           generate-id($context/ancestor::db:variablelist[1])
5041           =
5042           generate-id($target/parent::db:variablelist)">
5043     <xsl:text></xsl:text>
5044 </xsl:when>
5045 <!-- everything else except ranges, done below -->
5046 <xsl:when test="not($context/@endterm)">
5047     <xsl:text>\vref</xsl:text>
5048     <!-- defeat added space when preceding character
5049           was open-paren -->
5050     <xsl:if
5051       test="contains('({&lt;;',
5052             substring($context/
5053                       preceding-sibling::node()[1],
5054                       string-length(
5055                         $context/
5056                         preceding-sibling::node()[1])))">
5057       <xsl:text>*</xsl:text>
5058     </xsl:if>
5059 </xsl:when>
5060 <xsl:when test="$context/@endterm">
5061     <xsl:text></xsl:text>
5062 </xsl:when>
5063 <xsl:otherwise>
5064     <xsl:text>\ClassError{</xsl:text>
5065     <xsl:value-of select="$name"/>
5066     <xsl:text>}{xref attributes:</xsl:text>
5067     <xsl:for-each select="$context/@*">
5068       <xsl:text> </xsl:text>
5069       <xsl:value-of select="name()"/>
5070       <xsl:text>=</xsl:text>
5071       <xsl:value-of select="normalize-space(.)"/>
5072       <xsl:text>"</xsl:text>
5073     </xsl:for-each>
5074     <xsl:text>do not parse: contact author}</xsl:text>
5075 </xsl:otherwise>
5076 </xsl:choose>

```

Automate page range references, excluding the items already done; and enable a deep reference to a portion of a target structure.

```

5077 <!-- now the argument[s]: not wanted for list items
5078       done manually or for filename refs to extractable

```

```

5079         files in part/files or when source and target in
5080         same variablelist -->
5081     <xsl:if
5082         test="not(local-name($target)='listitem' and
5083             local-name($target/ancestor::*[1])='orderedlist')
5084             and
5085             not($context/ancestor::db:procedure
5086                 and
5087                 $target/parent::db:procedure)
5088             and
5089             not($target/ancestor::db:part[@xml:id='files'] and
5090                 $target/@xlink:href)
5091             and
5092             not(name($target)='varlistentry'
5093                 and
5094                 generate-id($context/ancestor::db:variablelist[1])
5095                 =
5096                 generate-id($target/parent::db:variablelist))">
5097         <xsl:if test="$context/@endterm">
5098             <xsl:text>\vrefrange</xsl:text>
5099         </xsl:if>
5100         <xsl:text>{</xsl:text>
5101         <xsl:value-of select="$idref"/>
5102         <xsl:text>}</xsl:text>
5103         <xsl:if test="$context/@endterm">
5104             <xsl:text>{</xsl:text>
5105             <xsl:value-of select="$context/@endterm"/>
5106             <xsl:text>}</xsl:text>
5107         </xsl:if>
5108     </xsl:if>
5109     <xsl:if test="$context/@xreflabel">
5110         <xsl:text> (</xsl:text>
5111         <xsl:apply-templates
5112             select="($target/child::*[local-name()=$context//
5113                 @xreflabel])[1]/node()"/>
5114         <xsl:text>)</xsl:text>
5115     </xsl:if>
5116 </xsl:for-each>
5117 </xsl:template>

```

db:pi Processing Instructions for L^AT_EX allow the inclusion of direct code for formatting fixups.

```

5118 <xsl:template match="processing-instruction('LaTeX')">
5119     <xsl:value-of select="."/>
5120 </xsl:template>
5121 <xsl:template match="processing-instruction('cp')">
5122     <xsl:variable name="args"
5123         select="tokenize(normalize-space(.),' ')/>
5124     <xsl:choose>
5125         <xsl:when test="$args[1]='include'">
5126             <xsl:apply-templates
5127                 select="document($args[2])/*[@xml:id=$args[3]]"/>
5128         </xsl:when>

```

```

5129     </xsl:choose>
5130 </xsl:template>

```

11.4.6 Special-purpose templates and utilities

db:annotation [db:annotation with [@annotations]]

For the UCC Thesis document class, this usage of <annotation> is quite different to normal, and allows to pull option declaration data from the specification tree held in the special "data" part. OBSOLETE soon, I hope.

```

5131 <!-- Former UCC bulk declarations now used for options
5132      Must have a para of explanation and the code in @annotations -->
5133 <xsl:template match="db:annotation[@annotations and @role='option']">
5134   <xsl:apply-templates select="db:para"/>
5135   <xsl:text>% \begin{CPKoption}{</xsl:text>
5136   <xsl:value-of select="@xreflabel"/>
5137   <xsl:text>}&#xa;% </xsl:text>
5138   <xsl:text>\begin</xsl:text>
5139   <xsl:text>{macrocode}&#xa;\DeclareOption{</xsl:text>
5140   <xsl:value-of select="@xreflabel"/>
5141   <xsl:text>}{%&#xa; </xsl:text>
5142   <xsl:value-of select="@annotations"/>
5143   <xsl:text>%&#xa;}&#xa;</xsl:text>
5144   <xsl:text>% </xsl:text>
5145   <xsl:text>\end</xsl:text>
5146   <xsl:text>{macrocode}</xsl:text>
5147   <xsl:text>&#xa;% \end{CPKoption}&#xa;</xsl:text>
5148 </xsl:template>
5149
5150 <xsl:template match="db:annotation/db:title">
5151   <xsl:text>% \subsection{</xsl:text>
5152   <xsl:apply-templates/>
5153   <xsl:text>}&#xa;</xsl:text>
5154 </xsl:template>

```

db:declopts OBSOLETE. This used <methodsynopsis> to express the L^AT_EX options for the very long lists of colleges, schools, departments, and their degrees, in an attempt to model the classes use by UCC for the thesis class.

```

5155 <xsl:template match="db:methodsynopsis" mode="declareoption">
5156   <!-- make a low-level title -->
5157   <xsl:text>% \paragraph{</xsl:text>
5158   <xsl:value-of
5159     select="normalize-space(db:methodparam[last()]
5160                           /db:parameter)"/>
5161   <!-- start a Code block for the option -->
5162   <xsl:text>}&#xa;% \begin{CPKoption}{</xsl:text>
5163   <xsl:value-of select="@xml:id"/>
5164   <xsl:text>}&#xa;% </xsl:text>
5165   <!-- now the descriptive text as a paragraph -->
5166   <!-- multiple parameters only for divisions, not degrees -->
5167   <!-- in document order, which is hierarchical top-down -->
5168   <xsl:for-each select="db:methodparam/db:parameter">

```

```

5169     <xsl:sort select="position()" order="descending"/>
5170     <xsl:if test="position()>1">
5171       <xsl:text> in </xsl:text>
5172     </xsl:if>
5173     <xsl:value-of select="normalize-space(.)"/>
5174     <xsl:if test="not(@role='degree')">
5175       <xsl:text> (</xsl:text>
5176       <xsl:value-of select="@role"/>
5177       <xsl:text>)</xsl:text>
5178     </xsl:if>
5179   </xsl:for-each>
5180   <xsl:text>; </xsl:text>
5181   <xsl:if test="not(db:methodparam/db:parameter/@role='degree')">
5182     <xsl:text>&#xa;% using the </xsl:text>
5183     <!-- langs in methodsynopsis/@language for babel -->
5184     <xsl:value-of select="normalize-space(db:methodname)"/>
5185     <xsl:text> citation format&#xa;% from the \verb`</xsl:text>
5186     <xsl:value-of select="db:methodparam[1]/db:initializer"/>
5187     <xsl:text>.bst` \BibTeX\ style</xsl:text>
5188     <xsl:if test="db:methodparam[1]/db:modifier">
5189       <xsl:text> with the </xsl:text>
5190       <xsl:for-each select="db:methodparam[1]/db:modifier">
5191         <xsl:if test="preceding-sibling::db:modifier">
5192           <xsl:text> and </xsl:text>
5193         </xsl:if>
5194         <xsl:text>\textsf{</xsl:text>
5195         <xsl:value-of select="."/>
5196         <xsl:text>}</xsl:text>
5197       </xsl:for-each>
5198       <xsl:text> package</xsl:text>
5199       <xsl:if test="count(db:methodparam[1]/db:modifier)>1">
5200         <xsl:text>s</xsl:text>
5201       </xsl:if>
5202     </xsl:if>
5203     <xsl:choose>
5204       <xsl:when test="contains(@xlink:href,'@')">
5205         <xsl:text> (confirmed by \url{</xsl:text>
5206         <xsl:value-of select="@xlink:href"/>
5207         <xsl:text>})</xsl:text>
5208       </xsl:when>
5209       <xsl:otherwise>
5210         <xsl:text> (unconfirmed)</xsl:text>
5211       </xsl:otherwise>
5212     </xsl:choose>
5213   </xsl:if>
5214   <!-- followed by the actual code for the declared option -->
5215   <xsl:text>.&#xa;% </xsl:text>
5216   <xsl:text>\begin</xsl:text>
5217   <xsl:text>{macrocode}&#xa;\DeclareOption{</xsl:text>
5218   <xsl:value-of select="@xml:id"/>
5219   <xsl:text>}{%&#xa;</xsl:text>
5220   <!-- order doesn't matter here,
5221       but the line length does (maxcodelen) -->
5222   <!-- so does duplication,

```

```

5223         because biblatex entries are replacements,
5224         not additions, but as \school etc use \gdef,
5225         there shouldn't be errors
5226         when they are duplicated -->
5227     <xsl:for-each select="db:methodparam/db:parameter">
5228         <xsl:text> \</xsl:text>
5229         <xsl:value-of select="@role"/>
5230         <xsl:if test="@remap=''">
5231             <xsl:text>[]</xsl:text>
5232         </xsl:if>
5233         <xsl:text>{</xsl:text>
5234         <!-- break at parenthesis if present -->
5235         <xsl:choose>
5236             <xsl:when test="contains(.,'(')">
5237                 <xsl:value-of
5238                     select="normalize-space(substring-before(.,'('))"/>
5239                 <xsl:text>&#xa;          (</xsl:text>
5240                 <xsl:value-of
5241                     select="normalize-space(substring-after(.,'('))"/>
5242             </xsl:when>
5243             <xsl:otherwise>
5244                 <xsl:value-of select="normalize-space(.)"/>
5245             </xsl:otherwise>
5246         </xsl:choose>
5247         <xsl:text>}&#xa;</xsl:text>
5248     </xsl:for-each>
5249     <!-- use found for \@usebib 4th param -->
5250     <xsl:if test="db:methodparam[1]/db:modifier or
5251                 db:methodparam[1]/db:initializer">
5252         <xsl:text> \@usebib</xsl:text>
5253         <!-- optional first param is package name[s] -->
5254         <xsl:if test="db:methodparam[1]/db:modifier">
5255             <xsl:text>[</xsl:text>
5256             <xsl:for-each select="db:methodparam[1]/db:modifier">
5257                 <xsl:if test="position()>1">
5258                     <xsl:text>,</xsl:text>
5259                 </xsl:if>
5260                 <xsl:value-of select="."/>
5261             </xsl:for-each>
5262             <xsl:text>]</xsl:text>
5263         </xsl:if>
5264         <!-- second param is bib[la]tex style name -->
5265         <xsl:text>{</xsl:text>
5266         <xsl:value-of select="db:methodparam[1]/db:initializer"/>
5267         <!-- third param is human-readable name of the style -->
5268         <xsl:text>}</xsl:text>
5269         <xsl:value-of select="db:methodname"/>
5270         <!-- fourth param is experimental signal for biblatex-->
5271         <xsl:text>}</xsl:text>
5272     <xsl:choose>
5273         <xsl:when test="db:methodparam/@role">
5274             <xsl:value-of select="db:methodparam/@role"/>
5275         </xsl:when>
5276         <xsl:otherwise>

```

```

5277         <xsl:text>\relax</xsl:text>
5278     </xsl:otherwise>
5279 </xsl:choose>
5280     <xsl:text>}&#xa;</xsl:text>
5281 </xsl:if>
5282 <!-- dept-specific languages one day
5283 <xsl:if test="@xml:lang">
5284     <xsl:text> \use@babel[</xsl:text>
5285     <xsl:value-of
5286         select="translate(normalize-space(@xml:lang),' ','')"/>
5287     <xsl:text>]</xsl:text>
5288     <xsl:text>british</xsl:text>
5289     <xsl:text>}&#xa;</xsl:text>
5290     <xsl:variable name="langsused">
5291         <xsl:value-of select="normalize-space(@xml:lang)"/>
5292     </xsl:variable>
5293     <xsl:if test="@audience">
5294         <xsl:for-each select="tokenize(@audience,' ')">
5295             <xsl:variable name="curtok" select="position()"/>
5296             <xsl:text> \def\</xsl:text>
5297             <xsl:value-of select="upper-case(.)"/>
5298             <xsl:text>{\foreignlanguage{</xsl:text>
5299             <xsl:value-of
5300                 select="tokenize($langsused,' ')"
5301                     [position()=$curtok]"/>
5302             <xsl:text>}}&#xa;</xsl:text>
5303         </xsl:for-each>
5304     </xsl:if>
5305 </xsl:if>
5306 -->
5307     <xsl:text>}&#xa;</xsl:text>
5308     <xsl:text>%    </xsl:text>
5309     <xsl:text>\end</xsl:text>
5310     <xsl:text>{macrocode}
5311 % \end{CPKoption}&#xa;</xsl:text>
5312 </xsl:template>

```

11.4.7 Text

text Parse all text being shipped out for special handling in two cases: *a*) shell script fragments in documentation (save space by removing initial newline and trailing space); and *b*) text of inline commands (and bibliography entries being handled by `db2bibtex.xml`): normalise spaces but preserve one leading and one trailing space if given. Otherwise, pass the string to the `dtxttext` template for handling of percent signs.

```

5313 <xsl:template match="text()">
5314     <xsl:choose>
5315         <xsl:when
5316             test="ancestor::db:programlisting[@language='bash']">
5317             <xsl:choose>
5318                 <!-- remove leading newlines from initial strings -->
5319                 <xsl:when test="count(preceding-sibling::text())=0
5320                     and starts-with(., '&#xa;')">

```

```

5321         <xsl:value-of select="substring(.,2)"/>
5322     </xsl:when>
5323     <!-- remove trailing space from terminals -->
5324     <xsl:when test="count(following-sibling::text())=0
5325         and matches(.,'\n[\s]*$')">
5326         <xsl:value-of select="replace(.,'\n[\s]*$', '')"/>
5327     </xsl:when>
5328     <!-- otherwise don't prefix or suffix -->
5329     <xsl:otherwise>
5330         <xsl:value-of select="."/>
5331     </xsl:otherwise>
5332 </xsl:choose>
5333 </xsl:when>
5334 <xsl:when
5335     test="ancestor::db:command or ancestor::db:biblioentry">
5336     <!-- replace a single leading WS token with a space -->
5337     <xsl:if test="starts-with(.,' ') or
5338         starts-with(.,'	') or
5339         starts-with(.,'
')">
5340         <xsl:text> </xsl:text>
5341     </xsl:if>
5342     <xsl:value-of select="normalize-space()"/>
5343     <!-- replace a single trailing WS token with a space -->
5344     <xsl:if test="substring(.,string-length())=' ' or
5345         substring(.,string-length())='	' or
5346         substring(.,string-length())='
'">
5347         <xsl:text> </xsl:text>
5348     </xsl:if>
5349 </xsl:when>
5350 <!-- otherwise check for percent prefixes -->
5351 <xsl:otherwise>
5352     <xsl:call-template name="dtxtext">
5353         <xsl:with-param name="text" select="."/>
5354     </xsl:call-template>
5355 </xsl:otherwise>
5356 </xsl:choose>
5357 </xsl:template>

```

11.4.8 Unhandled element types

default Catches everything not defined elsewhere and tells the user.

```

5358 <xsl:template match="*">
5359     <xsl:message>
5360         <xsl:text>Element default: </xsl:text>
5361         <xsl:for-each select="ancestor::*">
5362             <xsl:sort select="count(ancestor::*)" />
5363             <xsl:text></xsl:text>
5364             <xsl:value-of select="local-name()" />
5365         </xsl:for-each>
5366         <xsl:text></xsl:text>
5367         <xsl:value-of select="local-name()" />
5368         <xsl:text> ("</xsl:text>
5369         <xsl:value-of select="."/>

```

```

5370      <xsl:text>"</xsl:text>
5371    </xsl:message>
5372    <xsl:apply-templates/>
5373  </xsl:template>

```

11.5 Named templates

makepackages The biggest of these by far is *makepackages*, the procedure for creating a node-set of all the elements for the *AutoPackage* mechanism; that is, those for which an inclusion condition exists in `prepost.xml`.

11.5.1 Node-set of elements for *AutoPackage*

To collect these, we go through every `<step>`'s `<constructorsynopsis>` element and look in:

1. the *documentation* (the "doc" part), for any element whose name matches the current `<constructorsynopsis>`' `@condition` attribute;
2. the *metadata* (title, author, abstract, annotation) for elements matching as in [1] above;
3. the *licence* (eg LPPL) for elements matching as in [1].

the "doc" part itself is abused as a signal for when the `@condition` attribute is supplied but null, which means 'always include the package'; that is, its use stands duty for 'everywhere'. This also tests for the specific case of `@conformance` being "xelatex" while processing the entry for the *fontspec* package.

the *bibliography* is abused in a similar way when the `@arch` attribute holds the name of a bibliographic package (eg *biblatex*).

This used to be used for a style package in the `@xreflabel` attribute, but the use of old *BIBTEX* is slowly being deprecated.

11.5.1.1 Conditions for inclusion : Look in the `<constructorsynopsis>` element of each `<step>` (package) flagged for documentation in the `<prepackage>` section of the `prepost.xml` file.

```

5374  <!-- Called from the document root template -->
5375  <xsl:template name="makepackages">
5376    <!-- Go through every PRE-step in prepost.xml for DOC part
5377         and see if any of them match known conditions for use.
5378         We are creating $packages here.
5379         The CONTEXT here is <step> for the purposes of detection,
5380         but we create <seglistitem>s in the result tree
5381         so that they can be processed on a par with any author
5382         requests for packages in the document -->
5383    <xsl:for-each
5384      select="$prepost/db:procedure
5385             [@xml:id='prepackage']
5386             /db:step[contains(@condition,'doc')]">

```

11.5.1.2 Assemble the elements for each condition : These are the items in the list in [section 11.5.1 on the previous page](#).

`thisdoc` Note that `$thisdoc` is the current XML *ClassPack* document, so this variable
`elements` `$elements` will, in effect, contain all the printable content elements of the documentation⁷ matching the value of the current `@constructorsynopsis` attribute value from `prepost.xml`.

There are six nodesets to be combined, computed separately for ease of maintenance: the first four are each composed of the union of three nodesets, taken from the documentation text, the abstract, and the licence.

`elementmatch` 1. Selects packages from `prepost.xml` which specify an element type (only) from the author's document; for example the listings package would be selected if a `programlisting` element is present in the document:

```

5387      <!-- context here is db:step[@condition='doc'] -->
5388      <xsl:variable name="element-match"
5389        select="$thisdoc/db:book/db:part[@xml:id='doc']/
5390          descendant-or-self::*
5391          [for $c in current()/db:constructorsynopsis
5392            [@condition!='' and not(db:methodparam)]
5393            return (.[name()=$c/@condition])]
5394          |
5395          $thisdoc/db:book/db:info/
5396            (db:title|db:author|
5397            db:abstract|db:annotation)/
5398            descendant-or-self::*
5399            [for $c in current()/db:constructorsynopsis
5400              [@condition!='' and not(db:methodparam)]
5401              return (.[name()=$c/@condition])]
5402          |
5403          $licence/descendant-or-self::*
5404          [for $c in current()/db:constructorsynopsis
5405            [@condition!='' and not(db:methodparam)]
5406            return (.[name()=$c/@condition])]
5407      "/>

```

`attribmatch` 2. Selects packages from `prepost.xml` which specify an element type *with* an attribute from the author's document (regardless of the value of the attribute); for example the `graphicx` package would be selected if an `imagedata` element is present in the document with a `@fileref` attribute:

```

5408      <!-- context here is db:step[@condition='doc'] -->
5409      <xsl:variable name="attribute-match"
5410        select="$thisdoc/db:book/db:part[@xml:id='doc']/
5411          descendant-or-self::*
5412          [for $c in current()/db:constructorsynopsis
5413            [@condition!='' and
5414              db:methodparam[db:parameter and

```

⁷Documentation' here, and in the following sections on package selection, includes your package documentation; the Abstract and any note in an annotation element in the metadata; and the text of the License under which your package is issued.

```

5415         not(db:modifier)]]
5416     return (.[name()=$c/@condition and
5417         @*[name()=$c/db:methodparam/db:parameter]]))
5418 |
5419 $thisdoc/db:book/db:info/
5420     (db:title|db:author|
5421     db:abstract|db:annotation)/
5422     descendant-or-self::*
5423 [for $c in current()/db:constructorsynopsis
5424     [@condition!='' and
5425     db:methodparam[db:parameter and
5426     not(db:modifier)]]
5427     return (.[name()=$c/@condition and
5428         @*[name()=$c/db:methodparam/db:parameter]]))
5429 |
5430 $licence/descendant-or-self::*
5431 [for $c in current()/db:constructorsynopsis
5432     [@condition!='' and
5433     db:methodparam[db:parameter and
5434     not(db:modifier)]]
5435     return (.[name()=$c/@condition and
5436         @*[name()=$c/db:methodparam/db:parameter]]))
5437 "/>

```

attvalmatch 3. Selects packages from `prepost.xml` which specify an element type *with* an attribute containing a specific value from the author's document; for example the `fontspec` package would be selected if a `part` element is present in the document with a `@conformance` attribute set to the value `"xelatex"` or `"lualatex"`:

```

5438 <!-- context here is db:step[@condition='doc'] -->
5439 <xsl:variable name="attribute-value-match"
5440     select="$thisdoc/db:book/db:part[@xml:id='doc']/
5441     descendant-or-self::*
5442     [for $c in current()/db:constructorsynopsis
5443     [@condition!='' and
5444     db:methodparam[db:parameter and
5445     db:modifier]]
5446     return (.[name()=$c/@condition and
5447         @*[name()=$c/db:methodparam/db:parameter]=
5448         $c/db:methodparam/db:modifier]]))
5449 |
5450 $thisdoc/db:book/db:info/
5451     (db:title|db:author|
5452     db:abstract|db:annotation)/
5453     descendant-or-self::*
5454 [for $c in current()/db:constructorsynopsis
5455     [@condition!='' and
5456     db:methodparam[db:parameter and
5457     db:modifier]]
5458     return (.[name()=$c/@condition and
5459         @*[name()=$c/db:methodparam/db:parameter]=
5460         $c/db:methodparam/db:modifier]]))
5461 |
5462 $licence/descendant-or-self::*

```

```

5463         [for $c in current()/db:constructorsynopsis
5464             [@condition!='' and
5465             db:methodparam[db:parameter and
5466             db:modifier]]
5467         return (.[name()=$c/@condition and
5468             @*[name()=$c/db:methodparam/db:parameter]=
5469             $c/db:methodparam/db:modifier]])
5470     |
5471     $thisdoc/db:book
5472     [for $c in current()/db:constructorsynopsis
5473         [@condition!='' and
5474         db:methodparam[db:parameter and
5475         db:modifier]]
5476     return (.[name()=$c/@condition and
5477         @*[name()=$c/db:methodparam/db:parameter]=
5478         $c/db:methodparam/db:modifier]])
5479     "/>
5480     <!-- added db:book alone to handle matches
5481     to (eg) status="draft" -->

```

attvalptrmatch 4. Selects packages from `prepost.xml` which specify an element type with an IDREF or IDREFS attribute which links to an element of the specified type in the author's document; for example the `fmtcount` package would be selected if an `xref` element is present in the document with a `@linkend` attribute linking to a `varlistentry` element type (because the formatting of the link will need ordinal numbering):

```

5482     <!-- context here is db:step[@condition='doc'] -->
5483     <xsl:variable name="attribute-value-pointer"
5484         select="$thisdoc/db:book/db:part[@xml:id='doc']
5485             /descendant-or-self::*
5486             [name()='current()/db:constructorsynopsis/@condition]
5487             [for $m in
5488                 current()/db:constructorsynopsis/db:methodparam,
5489                 $e in tokenize(@*[name()=$m/db:funcparams])
5490                 return (id($e)[name()=$m/db:modifier]])
5491     |
5492     $thisdoc/db:book/db:info/
5493     (db:title|db:author|db:abstract|db:annotation)/
5494     descendant-or-self::*[name()='
5495     current()/db:constructorsynopsis/@condition]
5496     [for $m in
5497         current()/db:constructorsynopsis/db:methodparam,
5498         $e in tokenize(@*[name()=$m/db:funcparams])
5499         return (id($e)[name()=$m/db:modifier]])
5500     |
5501     $licence/descendant-or-self::*
5502     [name()='current()/db:constructorsynopsis/@condition]
5503     [for $m in
5504         current()/db:constructorsynopsis/db:methodparam,
5505         $e in tokenize(@*[name()=$m/db:funcparams])
5506         return (id($e)[name()=$m/db:modifier]])
5507     "/>

```

defaultmatch 5. Selects packages from `prepost.xml` which have a null value for `@condition`, meaning the package is to be selected every time:

```
5508      <!-- context here is db:step[@condition='doc'] -->
5509      <xsl:variable name="defaults"
5510          select="$thisdoc/db:book/db:part[@xml:id='doc']
5511              [current()/db:constructorsynopsis/@condition='']
5512          "/>
```

dependencymatch 6. Selects packages from `prepost.xml` which are a known dependency on a package specified by the author; for example the `url` package would be included if the `git` package has been requested, because `git` is known to require the `url` package with the `hyphens` option:

```
5513      <!-- context here is db:step[@condition='doc'] -->
5514      <!-- A dependency is signalled in prepost.xml when a package
5515           has a constructorsynopsis/@role AND a
5516           constructorsynopsis/methodname, meaning that IF a
5517           package in <methodname> (space-separated) is requested,
5518           OR a package in <methodname> is one that is also
5519           required BY DEFAULT, then the current package will
5520           become required, with the @role option (if non-null).
5521           So the package dependency MAY be signalled by using a
5522           null @role, meaning that no specific option is required,
5523           just the package.
5524      -->
5525      <xsl:variable name="dependencies"
5526          select="$thisdoc//db:constraintdef[@xml:id='docpackages']//
5527              db:seglistitem
5528              [db:seg=tokenize(normalize-space(
5529                  current()/db:constructorsynopsis[@role]
5530                  /db:methodname), ' ')]
5531              |
5532              current()[
5533                  $prepost/db:procedure[@xml:id='prepackage']/db:step
5534                  [db:constructorsynopsis/@condition='']
5535                  [@remap=tokenize(normalize-space(
5536                      current()/db:constructorsynopsis[@role]
5537                      /db:methodname), ' ')]
5538              "/>
```

elements Finally, create a nodeset of a union of all the above; these are the conditions by which the current package is going to be required.

```
5539      <xsl:variable name="elements"
5540          select="
5541              $element-match |
5542              $attribute-match |
5543              $attribute-value-match |
5544              $attribute-value-pointer |
5545              $defaults |
5546              $dependencies
5547          "/>
```

11.5.1.3 Create a result tree of packages : The list of required packages is expressed internally as a sequence of `<seglistitem>` elements so that it can be processed the same way as the list of packages requested manually by the author. An instance of the element will be added IFF there was at least one of the specified document elements found and recorded in one of the six component nodesets of the `$elements` variable created above.

We will need to echo to the console what packages were found and under what conditions, so we create a `$prepostno` variable which gives the section of `prepost.xml` and the sequence number of the step entry.

```

5548      <!-- REMINDER: context is still <step> in prepost,
5549              but the "elements" variable now contains every
5550              element FROM THE DOCUMENT which meets the conditions
5551              needed to justify the current package's inclusion.
5552              Process this ONLY IFF any elements at all were selected:
5553              this is where we create the result tree -->
5554      <xsl:if test="count($elements)>0">
5555          <!-- IDENTIFY THE CURRENT PACKAGE AND WHY [DE-]SELECTED -->
5556          <xsl:variable name="prepostno">
5557              <!-- this represents the number of the procedure in
5558                  prepost.xml (1=pre, 2=post) followed by the
5559                  (3-digit decimal) sequence of the position of
5560                  the package (the seglistitem element type),
5561                  so for example, 'dox' is 1.001 and fontawesome
5562                  is 1.127 -->
5563              <xsl:value-of
5564                  select="count(parent::db:procedure/
5565                          preceding-sibling::db:procedure) + 1"/>
5566              <xsl:text>.</xsl:text>
5567              <xsl:value-of
5568                  select="format-integer(
5569                      count(preceding-sibling::db:step)+1,
5570                      '999')"/>
5571              </xsl:variable>
5572          <!-- still in context <step> for the moment -->
5573          <xsl:variable name="thispackage" select="@remap"/>

```

In the following constructions, the attributes created by the [section 11.5.1.5 on page 227](#) named template reflect the values that are being persisted from the `prepost.xml` file.

Element-match: Package included because an element was used in the document.

```

5574      <!-- 1. ELEMENTS -->
5575      <xsl:if test="count($element-match)>0">
5576          <xsl:message>
5577              <xsl:value-of select="@remap"/>
5578              <xsl:text> (</xsl:text>
5579              <xsl:value-of select="$prepostno"/>
5580              <xsl:text>)</xsl:text>
5581              <xsl:text> element-match=</xsl:text>
5582              <xsl:value-of select="count($element-match)"/>

```

```

5583         <xsl:text> </xsl:text>
5584         <!-- element-match just needs element type name[s] -->
5585         <xsl:for-each-group select="$element-match"
5586             group-by="name()">
5587             <xsl:if test="position()>1">
5588                 <xsl:text>,</xsl:text>
5589             </xsl:if>
5590             <xsl:value-of select="current-grouping-key()"/>
5591         </xsl:for-each-group>
5592     </xsl:message>
5593     <xsl:call-template name="makeseg">
5594         <xsl:with-param name="arch" select="'elements'"/>
5595         <xsl:with-param name="prepostno" select="$prepostno"/>
5596     </xsl:call-template>
5597 </xsl:if>

```

Element and attribute match: Package included because an element with an attribute was used in the document.

```

5598     <!-- 2. ATTRIBUTES -->
5599     <xsl:if test="count($attribute-match)>0">
5600         <xsl:message>
5601             <xsl:value-of select="@remap"/>
5602             <xsl:text> (</xsl:text>
5603             <xsl:value-of select="$prepostno"/>
5604             <xsl:text>)</xsl:text>
5605             <xsl:text> attribute-match=</xsl:text>
5606             <xsl:value-of select="count($attribute-match)"/>
5607             <xsl:text> </xsl:text>
5608             <!-- attribute-match needs element/@attribute -->
5609             <xsl:for-each-group select="$attribute-match"
5610                 group-by="name()">
5611                 <xsl:if test="position()>1">
5612                     <xsl:text>,</xsl:text>
5613                 </xsl:if>
5614                 <xsl:value-of select="current-grouping-key()"/>
5615             <xsl:text></xsl:text>
5616             <!-- variable contains entries which are the
5617                 whole elements within which an attribute
5618                 match was found, so iterate over the
5619                 attributes as groups and only print the
5620                 one which matches -->
5621             <xsl:variable name="element-key"
5622                 select="current-grouping-key()"/>
5623             <xsl:variable name="attribs">
5624                 <xsl:for-each select="current-group()">
5625                     <xsl:for-each select="@*">
5626                         <att>
5627                             <xsl:value-of select="name()"/>
5628                         </att>
5629                     </xsl:for-each>
5630                 </xsl:for-each>
5631             </xsl:variable>
5632             <xsl:for-each-group select="$attribs/*"

```

```

5633                                     group-by="name()"/>
5634     <xsl:if
5635       test=".=
5636         $prepost/db:procedure/db:step
5637         [@remap=$thispackage]
5638         /db:constructorsynopsis
5639         [@condition=$element-key]
5640         /db:methodparam/db:parameter">
5641       <xsl:text>@</xsl:text>
5642       <xsl:value-of
5643         select="$prepost/db:procedure/db:step
5644         [@remap=$thispackage]
5645         /db:constructorsynopsis
5646         [@condition=$element-key]
5647         /db:methodparam/db:parameter"/>
5648     </xsl:if>
5649   </xsl:for-each-group>
5650 </xsl:for-each-group>
5651 </xsl:message>
5652 <xsl:call-template name="makeseg">
5653   <xsl:with-param name="arch" select="'attributes'"/>
5654   <xsl:with-param name="prepostno" select="$prepostno"/>
5655 </xsl:call-template>
5656 </xsl:if>

```

Element and attribute match with value: Package was included because an element with an attribute set to a value was used in the document.

```

5657 <!-- 3. ATTRIBUTE VALUES -->
5658 <xsl:if test="count($attribute-value-match)>0">
5659   <xsl:message>
5660     <xsl:value-of select="@remap"/>
5661     <xsl:text> (</xsl:text>
5662     <xsl:value-of select="$prepostno"/>
5663     <xsl:text>)</xsl:text>
5664     <xsl:text> attribute-value match=</xsl:text>
5665     <xsl:value-of select="count($attribute-value-match)"/>
5666     <xsl:text> </xsl:text>
5667     <xsl:for-each-group select="$attribute-value-match"
5668       group-by="name()">
5669       <xsl:if test="position()>1">
5670         <xsl:text>,</xsl:text>
5671       </xsl:if>
5672       <xsl:value-of select="current-grouping-key()"/>
5673       <xsl:text></xsl:text>
5674     <xsl:for-each select="current-group()">
5675       <xsl:variable name="matchgi" select="."/>
5676       <xsl:for-each
5677         select="$prepost/db:procedure/db:step
5678         [@remap=$thispackage]
5679         /db:constructorsynopsis
5680         [@condition=current-grouping-key()]
5681         /db:methodparam">
5682     <xsl:if

```

```

5683         test="$matchgi/@*
5684             [name()=current()/db:parameter]
5685             =current()/db:modifier">
5686         <xsl:text>@</xsl:text>
5687         <xsl:value-of select="db:parameter"/>
5688         <xsl:text>=</xsl:text>
5689         <xsl:value-of select="db:modifier"/>
5690         <xsl:text>:</xsl:text>
5691     </xsl:if>
5692 </xsl:for-each>
5693 </xsl:for-each>
5694 </xsl:for-each-group>
5695 </xsl:message>
5696 <xsl:call-template name="makeseg">
5697     <xsl:with-param name="arch" select="'attvals'"/>
5698     <xsl:with-param name="prepostno" select="$prepostno"/>
5699 </xsl:call-template>
5700 </xsl:if>

```

Element refers to another by IDREF: Package was included because an element in the document referenced by IDREF another element in the document known to require features from the package.

```

5701 <!-- 4. ATTRIBUTE VALUE POINTERS -->
5702 <xsl:if test="count($attribute-value-pointer)>0">
5703     <xsl:message>
5704         <xsl:value-of select="@remap"/>
5705         <xsl:text> (</xsl:text>
5706         <xsl:value-of select="$prepostno"/>
5707         <xsl:text>)</xsl:text>
5708         <xsl:text> attribute-value-pointer (</xsl:text>
5709         <xsl:value-of select="count($attribute-value-pointer)"/>
5710         <xsl:text>)</xsl:text>
5711         <xsl:for-each-group select="$attribute-value-pointer"
5712             group-by="name()">
5713             <xsl:if test="position()>1">
5714                 <xsl:text>,</xsl:text>
5715             </xsl:if>
5716             <xsl:value-of select="current-grouping-key()"/>
5717             <xsl:text></xsl:text>
5718             <xsl:variable name="elementname"
5719                 select="current-grouping-key()"/>
5720             <xsl:for-each
5721                 select="$prepost/db:procedure/db:step
5722                     [@remap=$thispackage]
5723                     /db:constructorsynopsis
5724                     [@condition=current-grouping-key()]
5725                     /db:methodparam">
5726                 <!-- any element in the document with an xml:id
5727                     pointed TO by one of the current group
5728                     which element is of the type specified -->
5729                 <xsl:if
5730                     test="count(current-group()/@*
5731                         [name()=current()/db:funcparams]

```

```

5732             [id()/name()=current()/db:modifier]]>0">
5733             <xsl:text>@</xsl:text>
5734             <xsl:value-of select="db:funcparams"/>
5735             <xsl:text>></xsl:text>
5736             <xsl:value-of select="db:modifier"/>
5737             <xsl:text>; </xsl:text>
5738         </xsl:if>
5739     </xsl:for-each>
5740 </xsl:for-each-group>
5741 </xsl:message>
5742 <xsl:call-template name="makeseg">
5743     <xsl:with-param name="arch" select="'attvalidrefs'"/>
5744     <xsl:with-param name="prepostno" select="$prepostno"/>
5745 </xsl:call-template>
5746 </xsl:if>

```

Requested package requires another (dependency): Package was included because it is known to be required by another package requested by the author.

```

5747 <!-- 5. DEPENDENCIES match seglistitems from user requests
5748      or from other packages included as defaults -->
5749 <xsl:if test="count($dependencies)>0">
5750     <xsl:message>
5751         <xsl:value-of select="@remap"/>
5752         <xsl:text> (</xsl:text>
5753         <xsl:value-of select="$prepostno"/>
5754         <xsl:text>)</xsl:text>
5755         <xsl:text> dependency match </xsl:text>
5756         <xsl:for-each-group select="$dependencies"
5757             group-by="name()">
5758             <xsl:if test="position()>1">
5759                 <xsl:text>,</xsl:text>
5760             </xsl:if>
5761             <xsl:value-of
5762                 select="db:constructorsynopsis/db:methodname"/>
5763         </xsl:for-each-group>
5764     </xsl:message>
5765     <xsl:call-template name="makeseg">
5766         <xsl:with-param name="arch" select="'dependencies'"/>
5767         <xsl:with-param name="prepostno" select="$prepostno"/>
5768     </xsl:call-template>
5769 </xsl:if>

```

Package included by default: Package was included because it is needed in all *ClassPack* classes and packages.

```

5770 <!-- 6. DEFAULT -->
5771 <xsl:if test="count($defaults)>0">
5772     <xsl:message>
5773         <xsl:value-of select="@remap"/>
5774         <xsl:text> (</xsl:text>
5775         <xsl:value-of select="$prepostno"/>
5776         <xsl:text>)</xsl:text>
5777         <xsl:text> default </xsl:text>
5778     <xsl:if test="$prepost/db:procedure[@xml:id='prepackage']

```

```

5779         /db:step[db:constructorsynopsis
5780         [contains(@remap,current()/@remap)]]"]>
5781     <xsl:text>will be omitted because of </xsl:text>
5782     <xsl:value-of
5783         select="$prepost/db:procedure[@xml:id='prepackage']
5784         /db:step[db:constructorsynopsis
5785         [contains(@remap,current()/@remap)]]/@remap"/>
5786     </xsl:if>
5787 </xsl:message>
5788 <xsl:call-template name="makeseg">
5789     <xsl:with-param name="arch" select="'defaults'"/>
5790     <xsl:with-param name="prepostno" select="$prepostno"/>
5791 </xsl:call-template>
5792 </xsl:if>
5793 </xsl:if>
5794 </xsl:for-each>

```

11.5.1.4 Packages explicitly included by the author : Packages named in the `<constraintdef>` element ID'd as "docpackages" and not set to "off" are positioned at the end of the list (numbered 9999) unless a @user level value sets this differently. The url and xcolor packages are excluded, as they will have been pre-loaded.

```

5795 <!-- 7. FINALLY, ADD PACKAGES REQUESTED BY THE AUTHOR
5796     in the document itself for use in the documentation.
5797     The list is accessed via the 'doc' pre packages,
5798     matching those mentioned in the author's document.
5799     This ensures that the context is <step> as earlier.
5800     The $authorloc variable points at the matching entry
5801     in the author's document so that options from BOTH
5802     $prepost AND the author's document can be merged. -->
5803 <xsl:for-each
5804     select="$prepost/
5805         db:procedure[@xml:id='prepackage']/
5806         db:step[contains(@condition,'doc')]
5807         [@remap=
5808             $thisdoc/db:book/db:info/db:cover
5809             /db:constraintdef[@xml:id='docpackages']
5810             /db:segmentedlist/db:seglistitem
5811             [not(db:seg/@condition='off')]/db:seg]">
5812     <xsl:variable name="prepostno">
5813         <xsl:value-of
5814             select="count(parent::db:procedure/
5815                 preceding-sibling::db:procedure) + 1"/>
5816         <xsl:text>.</xsl:text>
5817         <xsl:value-of
5818             select="format-integer(
5819                 count(preceding-sibling::db:step)+1,
5820                 '999')"/>
5821     </xsl:variable>
5822     <xsl:variable name="authorloc"
5823         select="$thisdoc/db:book/db:info/db:cover
5824             /db:constraintdef[@xml:id='docpackages']

```

```

5825         /db:segmentedlist/db:seglistitem
5826         [db:seg=current()/@remap]"/>
5827     <!-- These WERE flagged by making $prepostno 999 +
5828           the author's <seglistitem> but now with ordering
5829           in prepost.xml, this has reverted to 1 + -->
5830     <xsl:call-template name="makeseg">
5831       <xsl:with-param name="arch" select="'requests'"/>
5832       <!--
5833       <xsl:with-param name="prepostno">
5834         <xsl:text>999.</xsl:text>
5835       <xsl:value-of
5836         select="format-integer(
5837           count($authorloc/preceding-sibling::db:seglistitem)
5838             +1,'999')"/>
5839     </xsl:with-param>
5840     -->
5841     <xsl:with-param name="prepostno" select="$prepostno"/>
5842     <xsl:with-param name="authorloc" select="$authorloc"/>
5843   </xsl:call-template>
5844 </xsl:for-each>
5845 </xsl:template>
5846 <!-- end of makepackages -->

```

11.5.1.5 Constructing the list : Named template to assemble the data into a seglistitem format.

```

5847 <!-- called from within makepackages
5848       with context db:step from prepost.xml -->
5849 <xsl:template name="makeseg">
5850   <xsl:param name="arch"/>
5851   <xsl:param name="prepostno"/>
5852   <xsl:param name="authorloc"/>
5853   <!-- $authorloc is the <seglistitem> from the author's document
5854       where this package is being requested (requests only)
5855       OR from prepost.xml entry where the methodname contained
5856       a required package (doc only) -->
5857   <db:seglistitem arch="{ $arch}" userlevel="{ $prepostno}">
5858     <!-- add @condition to reflect an attribute match
5859         OR a package match for a dependency -->
5860     <xsl:choose>
5861       <xsl:when test="($arch='attributes' or $arch='attvals')
5862         and db:constructorsynopsis/@condition">
5863         <xsl:attribute name="condition"
5864           select="db:constructorsynopsis/@condition"/>
5865       </xsl:when>
5866       <xsl:when test="$arch='dependencies'">
5867         <xsl:attribute name="condition"
5868           select="current()/db:constructorsynopsis[@role]
5869             /db:methodname"/>
5870       </xsl:when>
5871     </xsl:choose>
5872     <!-- SOMETHING WRONG HERE: role being included when not
5873         needed eg hyphens on url -->

```

```

5874 <!-- specify dependency option as @role only when non-null
5875      (a null @role may have been used to signal a dependency
5876      but without the need for any specific option)... -->
5877 <xsl:if
5878   test="db:constructorsynopsis/@role
5879         and db:constructorsynopsis/@role!=''
5880         and $arch='dependencies'
5881         or (db:constructorsynopsis/db:methodname
5882             and db:constructorsynopsis/db:methodname=$authorloc)">
5883   <xsl:attribute name="role"
5884     select="db:constructorsynopsis/@role"/>
5885 </xsl:if>
5886 <!-- ...so when setting a dependency as a result of another
5887      package, the methodname must really have been used,
5888      so we check the author location -->
5889 <xsl:if
5890   test="db:constructorsynopsis/@role
5891         and $arch='dependencies'
5892         and db:constructorsynopsis/db:methodname
5893         and db:constructorsynopsis/db:methodname=$authorloc">
5894   <xsl:attribute name="audience"
5895     select="db:constructorsynopsis/db:methodname"/>
5896 </xsl:if>
5897 <xsl:if test="db:constructorsynopsis/@remap">
5898   <xsl:attribute name="remap"
5899     select="db:constructorsynopsis/@remap"/>
5900 </xsl:if>
5901 <!-- only add @conformance when it's an attribute
5902      or attval match -->
5903 <xsl:if
5904   test="($arch='attributes' or $arch='attvals')
5905         and
5906         db:constructorsynopsis/db:methodparam/db:parameter">
5907   <xsl:attribute name="conformance"
5908     select="db:constructorsynopsis/db:methodparam
5909             /db:parameter"/>
5910 </xsl:if>
5911 <!-- @annotations in <seglistitem> is DIFFERENT to
5912      @annotations in <seg> -->
5913 <xsl:if
5914   test="db:constructorsynopsis/db:methodparam/db:modifier">
5915   <xsl:attribute name="annotations"
5916     select="db:constructorsynopsis/db:methodparam
5917             /db:modifier"/>
5918 </xsl:if>
5919 <xsl:if
5920   test="db:constructorsynopsis/db:methodparam/db:funcparams">
5921   <xsl:attribute name="xreflabel"
5922     select="db:constructorsynopsis/db:methodparam
5923             /db:funcparams"/>
5924 </xsl:if>
5925 <db:seg>
5926   <xsl:if test="(@performance and $arch='dependencies') or
5927     @performance='required'">

```

```

5928     <xsl:attribute name="annotations" select="@performance"/>
5929 </xsl:if>
5930 <!-- $authorloc is missing EXCEPT when doing author-specified
5931 packages so here we create an @role attribute (options)
5932 if the @role is present (whether or not $authorloc is
5933 present) but we ADD any author-supplied options if
5934 $authorloc IS present (that is, if the author requested
5935 any options for a package that is already in $prepost
5936 with default options) -->
5937 <xsl:if test="$authorloc">
5938     <xsl:attribute name="security">
5939         <xsl:value-of select="normalize-space($authorloc)"/>
5940     </xsl:attribute>
5941 </xsl:if>
5942 <xsl:choose>
5943     <!-- @role only: use it -->
5944     <xsl:when test="@role and not($authorloc)">
5945         <xsl:attribute name="role">
5946             <xsl:value-of select="@role"/>
5947         </xsl:attribute>
5948     </xsl:when>
5949     <!-- no @role but $authorloc has a @role of its own -->
5950     <xsl:when test="$authorloc and
5951                     not(@role) and
5952                     $authorloc/db:seg/@role">
5953         <xsl:attribute name="role">
5954             <xsl:value-of select="$authorloc/db:seg/@role"/>
5955         </xsl:attribute>
5956     </xsl:when>
5957     <!-- @role and $authorloc and $authorloc has a
5958         @role of its own -->
5959     <xsl:when test="@role and
5960                     $authorloc and
5961                     $authorloc/db:seg/@role">
5962         <xsl:attribute name="role">
5963             <xsl:value-of select="@role | $authorloc/db:seg/@role"/>
5964         </xsl:attribute>
5965     </xsl:when>
5966     <!-- neither @role nor $authorloc means no @role needed
5967 <xsl:otherwise>
5968     <xsl:attribute name="security">
5969         <xsl:if test="@role">
5970             <xsl:value-of select="@role"/>
5971             <xsl:text> role</xsl:text>
5972         </xsl:if>
5973         <xsl:if test="$authorloc and $authorloc/db:seg/@role!=''">
5974             <xsl:value-of select="$authorloc/db:seg/@role"/>
5975             <xsl:text> authorloc</xsl:text>
5976         </xsl:if>
5977     </xsl:attribute>
5978 </xsl:otherwise>
5979 -->
5980 </xsl:choose>
5981 <xsl:if test="@revision">

```

```

5982         <xsl:attribute name="revision" select="@revision"/>
5983     </xsl:if>
5984     <xsl:if test="@conformance">
5985         <xsl:attribute name="conformance" select="@conformance"/>
5986     </xsl:if>
5987     <xsl:if test="@condition">
5988         <xsl:attribute name="condition" select="@condition"/>
5989     </xsl:if>
5990     <xsl:value-of select="@remap"/>
5991 </db:seg>
5992 </db:seglistitem>
5993 </xsl:template>

```

11.5.2 Other named templates

`makelisting` This is a tricky one: to handle the code listings both for DocTeX macrocode and for documentary listings in a consistent manner. This template gets invoked at various points in the handling of `<programlisting>` elements.

Start by getting the file type (extension) of any `@xlink:href` attribute.

```

5994 <!-- context is still programlisting -->
5995 <xsl:template name="makelisting">
5996     <xsl:variable name="fileext"
5997         select="tokenize(@xlink:href,'\.[position()=last()]')"/>
5998     <xsl:choose>

```

If the `@xlink:href` attribute was present, it means we need to read the content from the named external file, and use the `\lstinputlisting` command. The language argument always gets set first because it's compulsory (also useful because it means all subsequent arguments can be preceded by their comma). These are languages supported by the listings package, as amended by *Classpack* in `prepost.xml`.

```

5999         <xsl:when test="@xlink:href">
6000             <xsl:text>% \lstinputlisting[</xsl:text>
6001             <xsl:text>language=</xsl:text>
6002             <xsl:choose>
6003                 <xsl:when test="@language='LaTeXe'
6004                     or @language='LaTeX'
6005                     or not(@language)">
6006                 <xsl:text>{LaTeXe}</xsl:text>
6007             </xsl:when>
6008             <xsl:otherwise>
6009                 <xsl:value-of select="@language"/>
6010             </xsl:otherwise>
6011             </xsl:choose>

```

If we are using line-range marker labels (ie start and end) *and not* using fragments (generated by *chunk* (1), we need to specify the syntax of the range prefix and suffix (`&&` for TeX and BibTeX; a `<?cpdoc...?>` PI for XML and XSL; or a `#` for scripts).

```

6012         <xsl:if test="@startinglinenumber or @endinglinenumber">

```

```

6013     <xsl:if
6014         test="(not(number(@startinglinenumber))
6015             or
6016             not(number(@endinglinenumber)))
6017             and
6018             not(@conformance='frag'))">
6019     <xsl:choose>
6020         <xsl:when test="$fileext='tex'
6021             or
6022             $fileext='bib'">
6023             <xsl:text>,rangeprefix=\\%\\%\\ </xsl:text>
6024             <xsl:text>,rangesuffix=\\ \\%\\%</xsl:text>
6025         </xsl:when>
6026         <xsl:when test="$fileext='xml'
6027             or
6028             $fileext='xsl'
6029             or
6030             $fileext='xmap'">
6031             <xsl:text>,rangeprefix={\&lt;?cpdoc\ }</xsl:text>
6032             <xsl:text>,rangesuffix={?\>}</xsl:text>
6033         </xsl:when>
6034         <xsl:otherwise>
6035             <xsl:text>,rangeprefix=\\#\\#\\ </xsl:text>
6036             <xsl:text>,rangesuffix=\\ \\#\\#</xsl:text>
6037         </xsl:otherwise>
6038     </xsl:choose>
6039 </xsl:if>

```

Set the from and to boundaries, unless the file is being done in fragments.

```

6040     <xsl:if test="not(@conformance='frag'))">
6041     <!-- from -->
6042     <xsl:choose>
6043         <xsl:when test="@startinglinenumber">
6044             <xsl:choose>
6045                 <xsl:when
6046                     test="number(@startinglinenumber)">
6047                     <xsl:text>,firstline=</xsl:text>
6048                     <xsl:value-of
6049                         select="@startinglinenumber"/>
6050                 </xsl:when>
6051                 <xsl:otherwise>
6052                     <xsl:text>,linrange=</xsl:text>
6053                     <xsl:value-of
6054                         select="@startinglinenumber"/>
6055                 </xsl:otherwise>
6056             </xsl:choose>
6057         </xsl:when>
6058         <xsl:otherwise>
6059             <xsl:text>,firstline=1</xsl:text>
6060         </xsl:otherwise>
6061     </xsl:choose>
6062     <!-- to -->
6063     <xsl:choose>

```

```

6064         <xsl:when test="@endinglinenumber">
6065             <xsl:choose>
6066                 <xsl:when
6067                     test="number(@endinglinenumber)
6068                         or
6069                         number(@startinglinenumber)">
6070                     <xsl:text>,lastline=</xsl:text>
6071                     <xsl:value-of
6072                         select="@endinglinenumber"/>
6073                 </xsl:when>
6074                 <xsl:otherwise>
6075                     <xsl:text>-</xsl:text>
6076                     <xsl:value-of
6077                         select="@endinglinenumber"/>
6078                     <xsl:text>,includerangemarker=false</xsl:text>
6079                     <xsl:text>,showlines=false</xsl:text>
6080                 </xsl:otherwise>
6081             </xsl:choose>
6082         </xsl:when>
6083         <xsl:otherwise>
6084             <xsl:text></xsl:text>
6085         </xsl:otherwise>
6086     </xsl:choose>
6087     <!-- both -->
6088     <xsl:text>,numbers=left</xsl:text>
6089     <xsl:text>,numberstyle=\ttfamily</xsl:text>
6090     <xsl:text>\scriptsize\color{Blue}</xsl:text>
6091     <xsl:variable name="thisapp"
6092         select="(ancestor::db:chapter
6093             |
6094             ancestor::db:appendix)/@xml:id"/>
6095     <xsl:variable name="pth"
6096         select="count(preceding::db:programlisting
6097             [@xlink:href=current()/@xlink:href]
6098             [(ancestor::db:chapter|ancestor::db:appendix)
6099             /@xml:id=$thisapp])+1"/>
6100     <xsl:variable name="proglistgroup">
6101         <xsl:value-of select="$thisapp"/>
6102         <xsl:text>-</xsl:text>
6103         <xsl:value-of select="$pth"/>
6104     </xsl:variable>
6105     <xsl:choose>
6106         <!-- if this is the first of more than one -->
6107         <xsl:when
6108             test="$pth=1
6109                 and
6110                 count(following::db:programlisting
6111                     [@xlink:href=current()/@xlink:href]
6112                     [(ancestor::db:chapter|ancestor::db:appendix)
6113                     /@xml:id=$thisapp])>0">
6114             <xsl:text>,name=</xsl:text>
6115             <xsl:value-of select="$proglistgroup"/>
6116             <xsl:text>,firstnumber=1</xsl:text>
6117         </xsl:when>

```

```

6118             <xsl:otherwise>
6119                 <xsl:text>,name=</xsl:text>
6120                 <xsl:value-of select="$proglistgroup"/>
6121                 <xsl:text>,firstnumber=last</xsl:text>
6122             </xsl:otherwise>
6123         </xsl:choose>
6124     </xsl:if>
6125 </xsl:if>
6126 </xsl:when>
6127     <!-- still waiting for the closing square bracket -->

```

`makelisting` **Starting the listing environment.**

An extreme case to do first is to test for documenting the `lstlisting` itself (see the listings manual for details). There is a matching tag at the end of this template.

```

6128     <xsl:when test="contains(.,'\begin{lstlisting}')">
6129         <xsl:text>% \iffalse
6130 %&lt;*ignore>
6131 % \fi&#xa;</xsl:text>
6132         <xsl:text>\begin{listingsdoc}</xsl:text>
6133     </xsl:when>

```

Otherwise the start is similar but for the environment name.

```

6134     <!-- otherwise do a begin and end -->
6135     <xsl:otherwise>
6136         <xsl:text>% \iffalse
6137 %&lt;*ignore>
6138 % \fi&#xa;</xsl:text>
6139         <xsl:text>\begin{lstlisting}</xsl:text>
6140     </xsl:otherwise>
6141 </xsl:choose>

```

If there is *no* external file, specify the language first, for the same reasons as above.

```

6142     <xsl:if test="not(@xlink:href)">
6143         <xsl:text>language=</xsl:text>
6144         <xsl:choose>
6145             <xsl:when test="@language='LaTeXe'
6146                 or @language='LaTeX'
6147                 or not(@language)">
6148                 <xsl:text>{[LaTeX]TeX}</xsl:text>
6149             </xsl:when>
6150             <xsl:when test="@language='DocBook'">
6151                 <xsl:text>DocBook</xsl:text>
6152             </xsl:when>
6153             <xsl:otherwise>
6154                 <xsl:value-of select="@language"/>
6155             </xsl:otherwise>
6156         </xsl:choose>
6157     </xsl:if>

```

Now (for all cases except the documentation of the `lstlisting` itself, see above), go through the possible controls.

Whether the file is external or not, the `\basicstyle` (the default) is preset to `\footnotesize\color{Black}\ttfamily` in the `prepost.xml` configuration for listings.

- The `@wordsize` attribute is used to change the font size (either a size and baseline separated by a slash, eg "12/14") or a whole new `\basicstyle` definition);
- The `@arch="framed"` attribute signals a framed display (default is no frame);
- The `@remap` attribute changes the style (eg italics) of any keywords given in the `@annotations` attribute (below);
- The `@annotations` attribute specifies key words (or any `<user input>` content) needing special emphasis.
- The `@linenumbering` attribute specifies line numbering;
- The `@xml:id` attribute requests a label.

Finally, some adjustments are made to the width to allow for numbering, framing, etc.

```

6158     <xsl:if test="not(contains(.,'\begin{lstlisting}'))">
6159         <!-- font size -->
6160         <xsl:if test="@wordsize">
6161             <xsl:text>,breaklines=true</xsl:text>
6162             <xsl:text>,basicstyle=</xsl:text>
6163             <xsl:if test="@wordsize!=''">
6164                 <xsl:choose>
6165                     <xsl:when test="contains(@wordsize, '/')">
6166                         <xsl:text>\fontsize{</xsl:text>
6167                         <xsl:value-of
6168                             select="substring-before(@wordsize, '/')"/>
6169                         <xsl:text>}{</xsl:text>
6170                         <xsl:value-of
6171                             select="substring-after(@wordsize, '/')"/>
6172                         <xsl:text>}\selectfont</xsl:text>
6173                     </xsl:when>
6174                     <xsl:when test="number(@wordsize)">
6175                         <xsl:text>\fontsize{</xsl:text>
6176                         <xsl:value-of select="@wordsize"/>
6177                         <xsl:text>}{</xsl:text>
6178                         <xsl:value-of
6179                             select="number(@wordsize) * 1.2"/>
6180                         <xsl:text>}\selectfont</xsl:text>
6181                     </xsl:when>
6182                     <xsl:otherwise>
6183                         <xsl:value-of select="@wordsize"/>
6184                     </xsl:otherwise>
6185                 </xsl:choose>
6186             </xsl:if>

```

```

6187     <xsl:text>\color{Black}\ttfamily,</xsl:text>
6188 </xsl:if>
6189 <!-- default is unframed -->
6190 <xsl:if test="@arch='framed'">
6191     <xsl:text>,<frame=single,framesep=1em</xsl:text>
6192 </xsl:if>
6193 <!-- remap changes style of "emphasis" keywords -->
6194 <xsl:if test="@remap">
6195     <xsl:text>,<emphstyle=</xsl:text>
6196     <xsl:value-of select="@remap"/>
6197 </xsl:if>
6198 <!-- the emphasis keywords themselves -->
6199 <xsl:if test="@annotations or db:userinput">
6200     <xsl:text>,<emph={</xsl:text>
6201     <xsl:value-of select="@annotations"/>
6202     <xsl:if test="db:userinput">
6203         <xsl:for-each select="db:userinput">
6204             <xsl:if test="not(preceding-sibling::db:userinput
6205                 [. =current()/.] )">
6206                 <xsl:if test="(position()=1 and ../@annotations)
6207                     or
6208                     position()>1">
6209                     <xsl:text>,</xsl:text>
6210                 </xsl:if>
6211                 <xsl:value-of select="."/>
6212             </xsl:if>
6213         </xsl:for-each>
6214     </xsl:if>
6215     <xsl:text>}</xsl:text>
6216 </xsl:if>
6217 <!-- numbering -->
6218 <xsl:if test="@linenumbering='numbered'">
6219     <xsl:text>,<numbers=left</xsl:text>
6220     <xsl:text>,<numberstyle=\sffamily\footnotesize</xsl:text>
6221     <xsl:text>,<numbersep=2em</xsl:text>
6222 </xsl:if>
6223 <!-- label -->
6224 <xsl:if test="@xml:id">
6225     <xsl:text>,<label=</xsl:text>
6226     <xsl:value-of select="@xml:id"/>
6227 </xsl:if>
6228 <xsl:if test="@linenumbering='numbered' or @arch='framed'">
6229     <xsl:if test="@arch='framed'">
6230         <xsl:text>,<xrightmargin=1em</xsl:text>
6231     </xsl:if>
6232     <xsl:choose>
6233         <xsl:when
6234             test="@linenumbering='numbered' and @arch='framed'">
6235             <xsl:text>,<xleftmargin=3.5em</xsl:text>
6236         </xsl:when>
6237         <xsl:when test="@linenumbering='numbered'">
6238             <xsl:text>,<xleftmargin=2.5em</xsl:text>
6239         </xsl:when>
6240         <xsl:when test="@arch='framed'">

```

```

6241         <xsl:text>,xleftmargin=1em</xsl:text>
6242     </xsl:when>
6243 </xsl:choose>
6244 </xsl:if>
6245     <xsl:text>]</xsl:text>
6246 </xsl:if>

```

`makelisting` Then the filename or content

If a filename was specified, add it as an argument to the `\lstinputlisting` command generated above. If we are using fragments, the filename is catenated with a hyphen from the given filename and the `@startinglinenumber` string value, which is what the *chunk* (1) script creates.

```

6247 <xsl:choose>
6248 <!-- for an included file, here's the filename -->
6249 <xsl:when test="@xlink:href">
6250     <xsl:text>{</xsl:text>
6251     <xsl:choose>
6252         <xsl:when test="@conformance='frag'">
6253             <xsl:variable name="fn">
6254                 <xsl:choose>
6255                     <xsl:when test="contains(@xlink:href,'/')">
6256                         <xsl:value-of
6257                             select="tokenize(@xlink:href,'/')
6258                                 [position()=last()]" />
6259                     </xsl:when>
6260                     <xsl:otherwise>
6261                         <xsl:value-of select="@xlink:href" />
6262                     </xsl:otherwise>
6263                 </xsl:choose>
6264             </xsl:variable>
6265             <!-- add directory prefix if needed -->
6266             <xsl:if test="contains(@xlink:href,'/')">
6267                 <xsl:for-each
6268                     select="tokenize(@xlink:href,'/')
6269                         [not(position()=last())]">
6270                     <xsl:value-of select="." />
6271                     <xsl:text>/</xsl:text>
6272                 </xsl:for-each>
6273             </xsl:if>
6274             <xsl:value-of
6275                 select="substring-before($fn,'.')" />
6276             <xsl:text>-</xsl:text>
6277             <xsl:value-of select="@startinglinenumber" />
6278             <!--
6279             <xsl:text>-</xsl:text>
6280             <xsl:value-of select="@endinglinenumber" />
6281             -->
6282             <xsl:text>.</xsl:text>
6283             <xsl:value-of select="@conformance" />
6284             <!--
6285             <xsl:value-of select="$fileext" />
6286             -->

```

```

6287         </xsl:when>
6288         <xsl:otherwise>
6289             <xsl:value-of select="@xlink:href"/>
6290         </xsl:otherwise>
6291     </xsl:choose>
6292     <xsl:text>&#xa;</xsl:text>
6293 </xsl:when>

```

`lrtrim` Otherwise (no file), we use content, so we must start with a newline. The `lrtrim` named template handles the normalisation of the start and end of data, and also adds a 2-space indent if the content has any of the three environment end-commands specified.

```

6294     <xsl:otherwise>
6295         <xsl:text>&#xa;</xsl:text>
6296         <!-- output the content -->
6297         <xsl:call-template name="lrtrim">
6298             <xsl:with-param name="text" select="."/>
6299             <xsl:with-param name="indent">
6300                 <xsl:choose>
6301                     <xsl:when
6302                         test="contains(.,'\end{lstlisting}') or
6303                             contains(.,'\end{verbatim}') or
6304                             contains(.,'\end{Verbatim}')">
6305                         <xsl:text> </xsl:text>
6306                     </xsl:when>
6307                     <xsl:otherwise>
6308                         <xsl:text></xsl:text>
6309                     </xsl:otherwise>
6310                 </xsl:choose>
6311             </xsl:with-param>
6312         </xsl:call-template>

```

Finally, output the closing tags to match those added earlier.

```

6313     <xsl:choose>
6314         <xsl:when
6315             test="contains(.,'\begin{lstlisting}')">
6316             <xsl:text>&#xa;\end{listingsdoc}&#xa;</xsl:text>
6317             <xsl:text>% \iffalse
6318 %&lt;/ignore>
6319 % \fi&#xa;</xsl:text>
6320         </xsl:when>
6321         <xsl:otherwise>
6322             <xsl:text>&#xa;\end{lstlisting}&#xa;</xsl:text>
6323             <xsl:text>% \iffalse
6324 %&lt;/ignore>
6325 % \fi&#xa;</xsl:text>
6326         </xsl:otherwise>
6327     </xsl:choose>
6328 </xsl:otherwise>
6329 </xsl:choose>
6330 <xsl:if test="@conformance='frag'">
6331     <xsl:text>% \label{</xsl:text>

```

```

6332     <xsl:if test="number(@startinglinenumber)">
6333         <xsl:text>frag-</xsl:text>
6334     </xsl:if>
6335     <xsl:value-of select="@startinglinenumber"/>
6336     <xsl:text>}&#xa;</xsl:text>
6337 </xsl:if>
6338 </xsl:template>

```

`casestyle` This handles small caps for typefaces that don't have them (they must be flagged in `prepost.xml`). This is usually easier than trying to get L^AT_EX to do it. Create a variable holding the declared package entry in the document matching any known non-small-cap typefaces (eg Gotham, DejaVu) from `prepost.xml`. If there's a match, use a smaller size and all upper-case, otherwise use `\textsc` as normal.

```

6339 <xsl:template name="casestyle">
6340     <!-- disabled temporarily: just emits \textsc{...} -->
6341     <xsl:param name="text"/>
6342     <xsl:variable name="nosc">
6343         <xsl:for-each
6344             select="$prepost//db:step[@conformance='nosc']">
6345             <xsl:value-of
6346                 select="$thisdoc//db:constraintdef
6347                     [@xml:id='docpackages'
6348                     //db:seg[.='current()/@remap']"/>
6349         </xsl:for-each>
6350     </xsl:variable>
6351     <!--
6352     <xsl:choose>
6353         <xsl:when test="count($nosc)>0">
6354             <xsl:text>\SMC{</xsl:text>
6355             <xsl:value-of select="lower-case($text)"/>
6356             <xsl:text>}</xsl:text>
6357         </xsl:when>
6358         <xsl:otherwise>
6359             -->
6360             <xsl:text>\textsc{</xsl:text>
6361             <xsl:value-of select="lower-case($text)"/>
6362             <xsl:text>}</xsl:text>
6363         <!--
6364         </xsl:otherwise>
6365     </xsl:choose>
6366     -->
6367 </xsl:template>

```

AutoPackage

`nondeferred` *AutoPackage* provides for two modes of inclusion:

1. deferred — the `\RequirePackage` commands are not processed until (selectably) before or after a specific section of the class or package code, specified by the author;

2. non-deferred — all `\RequirePackage` commands are issued at the start of the class or package Preamble in the normal way.

This template provides for the second (default) mode. These are all called inline from the template which processes the document root, so their context node is `/`. The primary condition in the template checks that there are indeed some packages to process (those `<constraintdef>` packages for the current document type which are *not* re-use links, and whose `<seg>` subelement is non-null), so that it can output the DocTeX package tag, then processes each of them.

```

6368 <!-- CURRENTLY NON-OPERATIONAL -->
6369 <xsl:template name="nondeferredpackages">
6370   <!-- if there are any authorial non-null <seg> requests -->
6371   <xsl:if
6372     test="/db:book/db:info/db:cover/db:constraintdef
6373       [@xml:id=concat($filetype,'packages')]
6374       [not(@linkend)]
6375       /db:segmentedlist/db:seglistitem[db:seg!='']">
6376     <xsl:text>%&lt;package>%
6377 %&lt;package>% Packages invoked at start, before options
6378 %&lt;package>%#xa;</xsl:text>
6379     <!-- call packages on each seglistitem -->
6380     <xsl:for-each
6381       select="/db:book/db:info/db:cover/db:constraintdef
6382         [@xml:id=concat($filetype,'packages')]
6383         [not(@linkend)]
6384         /db:segmentedlist/db:seglistitem
6385         [db:seg!='']">
6386       <xsl:call-template name="packages">
6387         <xsl:with-param name="pkg" select="."/>
6388         <xsl:with-param name="dest" select="$filetype"/>
6389       </xsl:call-template>
6390     </xsl:for-each>
6391   </xsl:if>
6392 </xsl:template>

```

packages This template is the key to the *AutoPackage* mechanism. The `<seglistitem>` elements from `prepost.xml` and the user's document containing the names of the packages needed are passed into this template in the `$pkg` parameter, along with their destination (`$dest` being the current document type `"doc"`, `"cls"`, or `"pkg"`), whether this call is for deferred mode or non-deferred mode (`$mode`), and the element from which this template was called (`$loc`).

```

6393 <xsl:template name="packages">
6394   <xsl:param name="pkg"/>
6395   <xsl:param name="dest"/>
6396   <xsl:param name="cldbblocks"/>
6397   <xsl:param name="mode">
6398     <xsl:text></xsl:text>
6399   </xsl:param>
6400   <xsl:param name="loc"/>

```

If the package for loading in a class or style is explicitly mentioned in its own

<annotation> element (ie manually), it will get loaded there, so just output a comment here.

```

6401 <xsl:choose>
6402   <!-- OMIT BECAUSE THE PACKAGE IS ADDED MANUALLY -->
6403   <xsl:when
6404     test="($dest='cls' or $dest='sty')
6405         and
6406         $loc/ancestor::db:part[@xml:id='code']
6407         and
6408         ($loc/ancestor::db:chapter
6409           /descendant::db:annotation
6410            [@xreflabel=$pkg/db:seg][@role='package']
6411         or
6412         $loc/ancestor::db:appendix
6413           /descendant::db:annotation
6414            [@xreflabel=$pkg/db:seg][@role='package'])">
6415     <xsl:text>% Package \textsf{</xsl:text>
6416     <xsl:value-of select="$pkg/db:seg"/>
6417     <xsl:text>} omitted here (coded manually)</xsl:text>
6418     <xsl:text> on p.\pageref{ann-</xsl:text>
6419     <xsl:value-of select="$pkg/db:seg"/>
6420     <xsl:text>}. \par&#xa;%% Package \textsf{</xsl:text>
6421     <xsl:value-of select="$pkg/db:seg"/>
6422     <xsl:text>} omitted (coded manually)&#xa;</xsl:text>
6423   </xsl:when>
6424   <!-- no need now we have ifpackageloaded
6425   <xsl:when test="$dest='sty'
6426         and
6427         $thisdoc/db:book/@xlink:role
6428         and
6429         $pkg/db:seg='xcolor'">
6430     <xsl:text>% Package \textsf{</xsl:text>
6431     <xsl:value-of select="$pkg/db:seg"/>
6432     <xsl:text>} may get omitted to avoid clashes&#xa;</xsl:text>
6433   </xsl:when>
6434   -->

```

If this is [normal] ‘deferred’ mode, start the documentation block with any documentation for the package from the `prepost.xml` file (and if none, output a comment to that effect); then add any supplied by the user in the `@role` attribute. Repeat the documentation for the package from the `prepost.xml` file as a comment as well.

```

6435   <!-- OTHERWISE NORMAL INCLUSION -->
6436   <xsl:otherwise>
6437     <xsl:if test="$mode='deferred'">
6438       <xsl:text>% \begin{CPKpackage}{</xsl:text>
6439       <xsl:value-of select="$pkg/db:seg"/>
6440       <xsl:text>}&#xa;</xsl:text>
6441       <xsl:apply-templates
6442         select="$prepost/db:procedure
6443               [@xml:id='prepackage']
6444               /db:step[contains(@condition,$dest)]

```

```

6445             [@remap=$pkg/db:seg]/db:para"/>
6446 <!-- warn about package not in prepost -->
6447 <xsl:if
6448     test="count($prepost/db:procedure
6449                 [@xml:id='prepackage']
6450                 /db:step[contains(@condition,$dest)]
6451                 [@remap=$pkg/db:seg]/db:para)=0">
6452     <xsl:message>
6453         <xsl:text>WARNING: Package </xsl:text>
6454         <xsl:value-of select="$pkg/db:seg"/>
6455         <xsl:text> for </xsl:text>
6456         <xsl:value-of select="$filetype"/>
6457         <xsl:text> has no doc in prepost.xml</xsl:text>
6458     </xsl:message>
6459 </xsl:if>
6460 <!-- add comment from ClassPack document -->
6461 <xsl:if test="@role">
6462     <xsl:text>% </xsl:text>
6463     <xsl:value-of select="normalize-space(@role)"/>
6464     <xsl:text>&#xa;</xsl:text>
6465 </xsl:if>
6466 <!-- add explanation from prepost.xml if present -->
6467 <xsl:if test="count($prepost/db:procedure
6468                 [@xml:id='prepackage']
6469                 /db:step[contains(@condition,$dest)]
6470                 [@remap=$pkg/db:seg]/db:para)>0">
6471     <xsl:text>% \iffalse&#xa;%% </xsl:text>
6472     <xsl:value-of
6473         select="normalize-space(
6474             $prepost/db:procedure
6475             [@xml:id='prepackage']
6476             /db:step[contains(@condition,$dest)]
6477             [@remap=$pkg/db:seg]/db:para)"/>
6478     <xsl:text>&#xa;% \fi&#xa;</xsl:text>
6479 </xsl:if>
6480 <xsl:text>% </xsl:text>
6481 <xsl:text>\begin</xsl:text>
6482 <xsl:text>{macrocode}&#xa;</xsl:text>
6483 </xsl:if>

```

packages **Commands needed before package load**

Check if any special *preprocessing* is needed before invoking the package: this comes from `prepost.xml` in `<command>` elements in the `<constraintdef>` element in the "prepackage" section.

```

6484 <!-- output pre-commands -->
6485 <xsl:for-each
6486     select="$prepost/db:procedure
6487             [@xml:id='prepackage']
6488             /db:step[contains(@condition,$dest)]
6489             [@remap=$pkg/db:seg]
6490             /db:constraintdef/db:cmdsynopsis
6491             /db:command">
6492 <!-- check to escape internals in doc mode -->

```

```

6493         <xsl:if test="contains(., '@') and $mode=''">
6494             <xsl:text> \makeatletter&#xa;</xsl:text>
6495         </xsl:if>
6496         <xsl:text> </xsl:text>
6497         <xsl:value-of select="replace(., '^&#x9;', ', ')" />
6498         <xsl:text>&#xa;</xsl:text>
6499         <!-- end of escaping internals in doc mode -->
6500         <xsl:if test="contains(., '@') and $mode=''">
6501             <xsl:text> \makeatother&#xa;</xsl:text>
6502         </xsl:if>
6503     </xsl:for-each>

```

In classes and packages, we need to use `\RequirePackage`; in documentation but `\usepackage` in the class or style package code itself. In non-deferred mode in classes and packages, the command needs armour and `\RequirePackage` to be extractable.

```

6504         <xsl:if test="$dest='sty' and (
6505             $pkg/db:seg='xcolor'
6506             or
6507             $pkg/db:seg='hyperref'
6508             or
6509             $pkg/db:seg='url'
6510         )">
6511         <xsl:text>\IfPackageLoaded{</xsl:text>
6512         <xsl:value-of select="$pkg"/>
6513         <xsl:text>}{\relax}{%&#xa; </xsl:text>
6514         <!-- .true. requirepackage/usepackage goes here,
6515             .false. relax occurs further down -->
6516         </xsl:if>
6517         <xsl:choose>
6518             <xsl:when test="$mode='deferred'">
6519                 <xsl:text>\RequirePackage</xsl:text>
6520             </xsl:when>
6521             <xsl:when test="$dest=$filetype">
6522                 <xsl:text>%&lt;</xsl:text>
6523                 <xsl:value-of select="$doctype"/>
6524                 <xsl:text>>\RequirePackage</xsl:text>
6525             </xsl:when>
6526             <xsl:otherwise>
6527                 <xsl:text>\usepackage</xsl:text>
6528             </xsl:otherwise>
6529         </xsl:choose>

```

packages Package options

Options get output both for autodetected package and for those specified by the author, but not if the `<role>` attribute was explicitly set to null. An exclusion omits the **hyphens** option from the url package and the **svgnames** options from the xcolor package, *but a) hyphens* must now be a `\documentclass` option in the application document; and *b) svgnames* may need including if `@xlink:role` is specified on the master document root element `<book>`. If the `@condition` attribute is set to "only", just use the options specified by the author, and none from `prepost.xml`.

packages

TODO: Switch to remap if audience is biblatex and biblatex is specified.

```

6530      <xsl:if
6531        test="$pkg/db:seg/@role
6532          or
6533            $prepost/db:procedure[@xml:id='prepackage']/
6534            db:step[contains(@condition,$dest)]
6535            [@remap=$pkg/db:seg]/@role">
6536      <xsl:choose>
6537        <!-- elide null options -->
6538        <xsl:when test="$pkg/db:seg/@role and $pkg/db:seg/@role='
6539          and
6540          $prepost/db:procedure[@xml:id='prepackage']/
6541          db:step[contains(@condition,$dest)]
6542          [@remap=$pkg/db:seg]/@role=''">
6543          <xsl:text></xsl:text>
6544        </xsl:when>
6545        <!-- disallow options on package url disabled 2023-06-19
6546        <xsl:when test="$pkg/db:seg[.='url']">
6547          <xsl:text></xsl:text>
6548        </xsl:when>
6549        -->
6550        <!-- auto add svgnames option for xcolor
6551          when package requires itself in doc -->
6552        <xsl:when test="$pkg/db:seg='xcolor' and
6553          /db:book/@xlink:role and
6554          $filetype='sty'">
6555          <xsl:text>[svgnames]</xsl:text>
6556        </xsl:when>
6557        <!-- look no further -->
6558        <xsl:when
6559          test="$pkg/db:seg/@role
6560            and
6561            $pkg/db:seg/@condition='only'">
6562          <xsl:text></xsl:text>
6563          <xsl:value-of select="$pkg/db:seg/@role"/>
6564          <xsl:text></xsl:text>
6565        </xsl:when>

```

For all other cases, deduplicate the options specified in the author's package list and those found in `prepost.xml`.

```

6566      <xsl:otherwise>
6567        <!-- DEDUPLICATE -->
6568        <!-- string both sources together -->
6569        <xsl:variable name="reqs">
6570          <xsl:value-of
6571            select="normalize-space($pkg/db:seg/@role)"/>
6572          <xsl:text>,</xsl:text>
6573          <xsl:value-of
6574            select="normalize-space($prepost/db:procedure
6575              [@xml:id='prepackage']/db:step
6576              [contains(@condition,$dest)]

```

```

6577             [@remap=$pkg/db:seg]
6578             [contains('$pkg/db:seg/@role',@role)]
6579             /@role)"/>
6580     </xsl:variable>
6581     <!-- catch any null values -->
6582     <xsl:variable name="args">
6583       <xsl:for-each
6584         select="tokenize($reqs,',')[.!='']">
6585         <arg>
6586           <xsl:value-of select="normalize-space(.)"/>
6587         </arg>
6588       </xsl:for-each>
6589     </xsl:variable>
6590     <xsl:if test="count($args/arg[.!=''])>0">
6591       <xsl:text></xsl:text>
6592       <xsl:call-template name="rewrap">
6593         <xsl:with-param name="text">
6594           <xsl:for-each
6595             select="$args/arg[.!='']
6596               [not(preceding-sibling::arg=.)]">
6597             <xsl:if test="position()>1">
6598               <xsl:text>,</xsl:text>
6599             </xsl:if>
6600             <xsl:value-of select="."/>
6601           </xsl:for-each>
6602         </xsl:with-param>
6603       </xsl:call-template>
6604       <xsl:text>]</xsl:text>
6605     </xsl:if>
6606   </xsl:otherwise>
6607 </xsl:choose>
6608 </xsl:if>

```

packages Package name

Output the package name and version if needed: an author-specified date trumps any prepost.xml default.

```

6609     <!-- PACKAGE NAME -->
6610     <xsl:text></xsl:text>
6611     <xsl:value-of select="$pkg/db:seg"/>
6612     <xsl:text></xsl:text>
6613     <!-- POST-OPTIONS -->
6614     <xsl:if test="($pkg/db:seg/@version
6615       and
6616       $pkg/db:seg/@version!='')
6617       or
6618       $prepost/db:procedure
6619       [@xml:id='prepackage']/
6620       db:step[contains(@condition,$dest)]
6621       [@remap=$pkg/db:seg]/@revision">
6622       <xsl:text></xsl:text>
6623     <xsl:choose>
6624       <xsl:when test="$pkg/db:seg/@version[.!='']">
6625         <xsl:value-of

```

```

6626         select="translate(substring(
6627             $pkg/db:seg/@version,1,10),
6628             '-', '/')"/>
6629     </xsl:when>
6630     <xsl:when
6631         test="$prepost/db:procedure
6632             [@xml:id='prepackage']/
6633             db:step[contains(@condition,$dest)]
6634             [@remap=$pkg/db:seg]
6635             [@role=$pkg/db:seg/@role]/@revision">
6636         <xsl:value-of
6637             select="translate(substring(
6638                 $prepost/db:procedure
6639                 [@xml:id='prepackage']/db:step
6640                 [contains(@condition,$dest)]
6641                 [@remap=$pkg/db:seg]
6642                 [@role=$pkg/db:seg/@role]
6643                 /@revision,1,10),'-', '/')"/>
6644     </xsl:when>
6645 </xsl:choose>
6646 <xsl:text>]</xsl:text>
6647 </xsl:if>

```

packages **Comments**

Comments only get appended in non-deferred mode; deferred ones were added earlier as text.

```

6648     <xsl:if test="$pkg/@role and $mode=''">
6649         <xsl:text>% </xsl:text>
6650         <xsl:value-of
6651             select="normalize-space($pkg/@role)"/>
6652         <xsl:if test="@userlevel">
6653             <xsl:text> (</xsl:text>
6654             <xsl:value-of select="@userlevel"/>
6655             <xsl:text>)</xsl:text>
6656         </xsl:if>
6657     </xsl:if>
6658     <xsl:text>%&#xa;</xsl:text>
6659     <!-- complete the enclosure of ifpackageloaded -->
6660     <xsl:if test="$dest='sty' and
6661         ($pkg/db:seg='xcolor'
6662         or $pkg/db:seg='hyperref'
6663         or $pkg/db:seg='url')">
6664         <xsl:text>}&#xa;</xsl:text>
6665     </xsl:if>

```

packages **Commands needed after package load**

Check if any corresponding postprocessing is needed, escaping @ signs if needed as for the preprocessing.

```

6666     <!-- output post-commands -->
6667     <xsl:if test="not($pkg/db:seg/@condition='only')">
6668         <!-- go through any supplied commands

```

```

6669         that are not excluded -->
6670     <xsl:for-each
6671         select="$prepost/db:procedure
6672             [@xml:id='postpackage']/db:step
6673             [contains(@condition,$dest)]
6674             [@remap=$pkg/db:seg]
6675             /db:constraintdef/db:cmdsynopsis
6676             /db:command">
6677     <!-- exclude commands because of another package -->
6678     <xsl:if test="not(current()/@conformance=$cmdblocks)">
6679         <xsl:variable name="matchconditions"
6680             select="$prepost/db:procedure
6681                 [@xml:id='prepackage']
6682                 /db:step
6683                 [contains(@condition,$dest)]
6684                 [@remap=$pkg/db:seg]
6685                 /db:constructorsynopsis"/>
6686     <!-- load any command without an xml:id
6687         OR
6688         with an xml:id matching a condition from
6689         the prepost section (ie programlisting
6690         with a language whose name matches this
6691         xml:id. Silent if no matches. -->
6692     <xsl:if
6693         test="not(@xml:id)
6694             or
6695             @xml:id=$thisdoc/*
6696             [name()=$matchconditions/@condition]
6697             /@*
6698             [name()=$matchconditions
6699             /db:methodparam/db:parameter]">
6700         <xsl:if test="contains(.,'@') and $mode=''">
6701             <xsl:text> \makeatletter&#xa;</xsl:text>
6702         </xsl:if>
6703         <xsl:text> </xsl:text>
6704         <xsl:value-of
6705             select="replace(.,'^&#x9;','')"/>
6706         <xsl:text>&#xa;</xsl:text>
6707         <xsl:if test="contains(.,'@') and $mode=''">
6708             <xsl:text> \makeatother&#xa;</xsl:text>
6709         </xsl:if>
6710     </xsl:if>
6711 </xsl:if>
6712 </xsl:for-each>
6713 <!-- special for programlisting and the
6714     listing package: iterate over languages -->
6715 <xsl:if test="$pkg/db:seg='listings'">
6716     <xsl:for-each-group group-by="@language"
6717         select="$thisdoc
6718             //(db:programlisting|db:literallayout)">
6719         <xsl:text> \lstloadlanguages{</xsl:text>
6720         <xsl:value-of select="current-grouping-key()"/>
6721         <xsl:text>}&#xa;</xsl:text>
6722     </xsl:for-each-group>

```

```

6723         <xsl:text> \lstloadlanguages{dummy}&#xa;</xsl:text>
6724     </xsl:if>
6725 </xsl:if>

```

In deferred mode, terminate the code and output any postpackage documentation.

```

6726     <xsl:if test="$mode='deferred'">
6727         <xsl:text>% </xsl:text>
6728         <xsl:text>\end</xsl:text>
6729         <xsl:text>{macrocode}&#xa;</xsl:text>
6730         <!-- post-package comments from prepost.xml -->
6731         <xsl:apply-templates
6732             select="$prepost/db:procedure
6733                 [@xml:id='postpackage'
6734                 /db:step[contains(@condition,$dest)]
6735                 [@remap=current()/db:seg]/db:para"/>
6736         <xsl:text>% \end{CPKpackage}&#xa;</xsl:text>
6737     </xsl:if>

```

packages Self-loading

Finally, see if the package we are documenting needs itself loading for the documentation (eg for examples). This only applies in non-deferred mode.

```

6738     <xsl:if test="$mode='
6739         and
6740         $dest='doc'
6741         and
6742         $doctype='package'
6743         and
6744         /db:book/@role">
6745         <xsl:text>% \usepackage</xsl:text>
6746         <xsl:if test="/db:book/@role!=''">
6747             <xsl:text>[</xsl:text>
6748             <xsl:value-of select="/db:book/@role"/>
6749             <xsl:text>]</xsl:text>
6750         </xsl:if>
6751         <xsl:text>{</xsl:text>
6752         <xsl:value-of select="$docname"/>
6753         <xsl:text>}% added by spec.&#xa;</xsl:text>
6754     </xsl:if>
6755 </xsl:otherwise>
6756 </xsl:choose>
6757 </xsl:template>

```

rewrap Rewrapping is designed for a long options list in `\RequirePackage`. The `$text` parameter is a comma-separated vector or arguments; the `$maxlen` parameter is the maximum line length to output: the default is 32 but the value becomes 50 after the first line.

```

6758     <xsl:template name="rewrap">
6759         <xsl:param name="text"/>
6760         <xsl:param name="maxlen">

```

```

6761     <xsl:text>32</xsl:text>
6762 </xsl:param>
6763 <xsl:choose>
6764     <!-- if text is an overlong string
6765           but contains a comma -->
6766     <xsl:when
6767         test="string-length($text)>$maxlen
6768             and contains($text,',')">
6769         <!-- see if there is a comma
6770               within the first $maxlen chars -->
6771         <xsl:variable name="prelastcomma">
6772             <xsl:choose>
6773                 <xsl:when
6774                     test="contains(substring($text,1,$maxlen),
6775                                   ',')">
6776                     <xsl:analyze-string
6777                         select="substring($text,1,$maxlen)"
6778                         regex="^(.*)([^\,]*)$"
6779                         <xsl:matching-substring>
6780                             <xsl:value-of
6781                                 select="concat(regex-group(1),',' '/')"/>
6782                         </xsl:matching-substring>
6783                         <xsl:non-matching-substring>
6784                             <xsl:text>NO ", " IN SUBSTRING</xsl:text>
6785                         </xsl:non-matching-substring>
6786                     </xsl:analyze-string>
6787                 </xsl:when>
6788                 <!-- No comma in range,
6789                       test whole string -->
6790                 <xsl:otherwise>
6791                     <xsl:analyze-string select="$text"
6792                         regex="^(.*)([^\,]*)$"
6793                         <xsl:matching-substring>
6794                             <xsl:value-of
6795                                 select="concat(regex-group(1),',' '/')"/>
6796                         </xsl:matching-substring>
6797                         <xsl:non-matching-substring>
6798                             <xsl:text>NO ", " IN STRING</xsl:text>
6799                         </xsl:non-matching-substring>
6800                     </xsl:analyze-string>
6801                 </xsl:otherwise>
6802             </xsl:choose>
6803         </xsl:variable>
6804         <xsl:value-of select="$prelastcomma"/>
6805     </xsl:choose>
6806     <xsl:when
6807         test="ancestor::db:constraintdef
6808             [@xml:id=concat($filetype,'packages')]">
6809         <xsl:text>&#xa; </xsl:text>
6810     </xsl:when>
6811     <xsl:otherwise>
6812         <xsl:text>&#xa; </xsl:text>
6813     </xsl:otherwise>
6814 </xsl:choose>

```

```

6815     <xsl:call-template name="rewrap">
6816       <xsl:with-param name="text"
6817         select="normalize-space(
6818           substring-after(
6819             $text,$prelastcomma))"/>
6820     <xsl:with-param name="maxlen">
6821       <xsl:text>50</xsl:text>
6822     </xsl:with-param>
6823   </xsl:call-template>
6824 </xsl:when>
6825 <xsl:otherwise>
6826   <xsl:value-of select="$text"/>
6827 </xsl:otherwise>
6828 </xsl:choose>
6829 </xsl:template>

```

makechapapp This template handles the code needed to set up a chapter or appendix. It gets called from the chapter and appendix templates, which pass the variables

chapter
appendix
thisdocfullname
thisdocname
thisdoctype
thisdocfulltype
thisdocid

- \$thisdocfullname, which is the document name itself, unless there is an attribute @xlink:href, in which case that's the filename;
- \$thisdocname, which is the full name without the filetype;
- \$thisdoctype, which is the filetype;
- \$thisdocfulltype, which is the name Class, Package, or File;
- \$thisdocid, which is the <ID> of the chapter or appendix, or (if none) the <ID> of the document.

The variables will have no meaning in included external files like the licence documentation.

```

6830 <xsl:template name="makechapapp">
6831   <!-- output contents of chapter or appendix -->
6832   <!-- full output file name -->
6833   <xsl:variable name="thisdocfullname">
6834     <xsl:choose>
6835       <xsl:when test="@xlink:href">
6836         <xsl:value-of
6837           select="tokenize(@xlink:href,'/')
6838             [position()=last()]" />
6839       </xsl:when>
6840       <xsl:otherwise>
6841         <xsl:value-of select="$docname"/>
6842         <xsl:text>.</xsl:text>
6843         <xsl:value-of select="$filetype"/>
6844       </xsl:otherwise>
6845     </xsl:choose>
6846   </xsl:variable>
6847   <xsl:variable name="thisdocname"
6848     select="tokenize($thisdocfullname,'\.')[1]" />
6849   <xsl:variable name="thisdoctype"
6850     select="tokenize($thisdocfullname,'\.')
6851       [position()=last()]" />
6852   <xsl:variable name="thisdocfulltype">

```

```

6853     <xsl:choose>
6854       <xsl:when test="$thisdoctype='cls'">
6855         <xsl:text>Class</xsl:text>
6856       </xsl:when>
6857       <xsl:when test="$thisdoctype='sty'">
6858         <xsl:text>Package</xsl:text>
6859       </xsl:when>
6860       <xsl:otherwise>
6861         <xsl:text>File</xsl:text>
6862       </xsl:otherwise>
6863     </xsl:choose>
6864   </xsl:variable>
6865   <xsl:variable name="thisdocid">
6866     <xsl:choose>
6867       <xsl:when test="@xml:id">
6868         <xsl:value-of select="@xml:id"/>
6869       </xsl:when>
6870       <xsl:otherwise>
6871         <xsl:value-of select="/db:book/@xml:id"/>
6872       </xsl:otherwise>
6873     </xsl:choose>
6874   </xsl:variable>

```

Documentation is simply passed through with no other action.

```

6875   <!-- choices are: 'doc', 'code', and otherwise -->
6876   <xsl:choose>
6877     <!-- EXTRACT ALL THE DOCUMENTATION -->
6878     <xsl:when test="parent::db:part[@xml:id='doc']">
6879       <xsl:apply-templates/>
6880     </xsl:when>

```

makechapapp **Code**

Extractable T_EX files need a double comment character at the start of the lines that will be included in the file preamble, and need to be tagged with their own name and the file metadata. This means resetting the line counter now, not when the code proper starts.

makechapapp **TODO: Does this really apply only to 'main' chapters?**

In the "code" part, for the main chapter, any preliminary material (preceding any <sect 1>) needs to be output here, because the rest is done after auto-initialisation — provided it does not include any code at that stage (if it does, it'll get done later).

```

6881     <xsl:when test="parent::db:part[@xml:id='code']">
6882       <!-- CODE COUNTER NEEDS TO BE RESET to 0 here -->
6883       <xsl:if test="$thisdoctype='dtd' or
6884                   $thisdoctype='xml' or
6885                   $thisdoctype='xsl' or
6886                   $thisdoctype='awk' or
6887                   $thisdoctype='sh' or
6888                   $thisdoctype='cp' or

```

```

6889             $thisdoctype='bash'")
6890     <xsl:text>% \setcounter{CodelineNo}{0}</xsl:text>
6891     <xsl:text>&#xa;</xsl:text>
6892 </xsl:if>
6893 <!-- OUTPUT ANYTHING BEFORE THE FIRST <sect1>
6894      (usually some kind of preamble or explanation)
6895      provided there is NO code -->
6896 <xsl:if test="not(db:sect1[1]/preceding-sibling::*/*
6897      descendant-or-self::db:programlisting)">
6898     <xsl:apply-templates
6899       select="db:sect1[1]/preceding-sibling::*"/>
6900     <!-- if there is no sect1, or if there IS code,
6901          this does nothing -->
6902 </xsl:if>
6903 <!-- if there is a programlisting and an output
6904      filename and an ID, start with a comment. -->
6905 <xsl:if test="ancestor-or-self::db:appendix
6906      [descendant::db:programlisting
6907      and @xlink:href
6908      and @xml:id]">
6909     <!-- use sentinel to enclose the comment of
6910          identity in the right format -->
6911     <xsl:call-template name="startappcomment">
6912       <xsl:with-param name="scriptflag">
6913         <xsl:choose>
6914           <xsl:when test="@arch='script' and @userlevel='bash'">
6915             <xsl:text>wait</xsl:text>
6916           </xsl:when>
6917           <xsl:otherwise>
6918             <xsl:text>no</xsl:text>
6919           </xsl:otherwise>
6920         </xsl:choose>
6921       </xsl:with-param>
6922     </xsl:call-template>
6923 </xsl:if>

```

The auto-initialisation for classes and packages starts with a documentary explanation of what we are doing. For extracted appendices, it resets the code line counter to zero.

```

6924     <xsl:if test="$thisdoctype!='dtd' and
6925             $thisdoctype!='xml' and
6926             $thisdoctype!='xsl' and
6927             $thisdoctype!='awk' and
6928             $thisdoctype!='sh' and
6929             $thisdoctype!='cp' and
6930             $thisdoctype!='bash'">
6931     <xsl:text>% \subsection{Auto-initialisation}</xsl:text>
6932     <xsl:text>\label{</xsl:text>
6933     <xsl:value-of select="@xml:id"/>
6934     <xsl:text>:autoinit}>
6935 % This section is added automatically by \textit{ClassPack}
6936 % as a preamble to all classes and style packages.
6937 % For details see the \textsf{ltxdoc} package documentation.

```

```

6938 % \par\smallskip
6939 % &#xa;</xsl:text>
6940     <!-- reset the counter if this is being done
6941           from an <appendix>, because the counter
6942           will still show the last line of the main
6943           (<chapter>) block of code -->
6944     <xsl:if test="name()='appendix'">
6945       <!-- it should probably be reset anyway -->
6946       <xsl:text>% \setcounter{CodelineNo}{0}</xsl:text>
6947       <xsl:text>&#xa;</xsl:text>
6948     </xsl:if>

```

Because we are auto-generating some code before the actual code, starts, we need the code presented as a fake in the form of a table, to allow DocTeX to fulfil the documentation requirement to start numbering at the right line (we add three to the counter).

makechapapp **TODO: For ancillaries, should be their own version**

```

6949     <!-- working on ways to make this use lstlisting
6950           as it doesn't need macrocode, having been tagged
6951           <*package> or <*class> earlier -->
6952     <xsl:text>% \begingroup\color{DarkRed}</xsl:text>
6953     <xsl:text>\footnotesize&#xa;% \leavevmode</xsl:text>
6954     <!-- removed leading \enspace as it now interferes
6955           with the alignment of the first line 2023-05-15.
6956           Put it back 2023-05-16 -->
6957     <xsl:text>\enspace{\scriptsize1}\quad</xsl:text>
6958     <xsl:text>\ttfamily\textbackslash </xsl:text>
6959     <xsl:text>NeedsTeXFormat\{</xsl:text>
6960     <xsl:value-of select="//db:book/@conformance"/>
6961     <xsl:text>\}[</xsl:text>
6962     <xsl:value-of
6963       select="translate(/db:book/@condition,'-','/')"/>
6964     <xsl:text>]]\&#xa;% \leavevmode</xsl:text>
6965     <xsl:text>\enspace{\scriptsize2}\quad</xsl:text>
6966     <xsl:text>\ttfamily\textbackslash </xsl:text>
6967     <xsl:text>Provides</xsl:text>
6968     <xsl:value-of select="$thisdocfulltype"/>
6969     <xsl:text>\{</xsl:text>
6970     <xsl:value-of select="$thisdocname"/>
6971     <xsl:text>\}[</xsl:text>
6972     <xsl:value-of select="translate($date,'-','/')"/>
6973     <xsl:text> v</xsl:text>
6974     <xsl:value-of select="//db:book/@version"/>
6975     <xsl:text>.</xsl:text>
6976     <xsl:value-of select="//db:book/@revision"/>
6977     <xsl:text>}\&#xa;% \leavevmode</xsl:text>
6978     <xsl:text>\enspace{\scriptsize3}\quad</xsl:text>
6979     <xsl:text>\ttfamily </xsl:text>
6980     <xsl:choose>
6981       <xsl:when
6982         test="//db:part[@xml:id='code']/db:title">

```

```

6983         <xsl:value-of
6984             select="normalize-space(replace(
6985                 //db:part[@xml:id='code']/db:title,
6986                 '\\LaTeX\\{\\}', 'LaTeX'))"/>
6987     </xsl:when>
6988     <xsl:when test="name()='appendix'">
6989         <xsl:value-of
6990             select="normalize-space(replace(db:title,
6991                 '\\LaTeX\\{\\}', 'LaTeX'))"/>
6992     </xsl:when>
6993     <xsl:otherwise>
6994         <xsl:value-of
6995             select="normalize-space(/db:book/db:info
6996                 /db:title)"/>
6997     </xsl:otherwise>
6998 </xsl:choose>
6999 <xsl:text>]]\\endgroup
7000 % \setcounter{CodelineNo}{3}&#xa;</xsl:text>

```

The actual startup code can now be output in normal DocTeX form. This ends the initialisation phase.

```

7001 <!-- APPENDICES ONLY -->
7002 <xsl:if
7003     test="ancestor-or-self::db:appendix and (
7004         ancestor-or-self::db:appendix/@userlevel='tex' or
7005         ancestor-or-self::db:appendix/@userlevel='sty' or
7006         ancestor-or-self::db:appendix/@userlevel='cls'
7007     )">
7008     <xsl:text>% \iffalse&#xa;%&lt;</xsl:text>
7009     <xsl:value-of select="$thisdocid"/>
7010     <xsl:text>>\NeedsTeXFormat{LaTeX2e}</xsl:text>
7011     <xsl:text>[2015/01/01]&#xa;</xsl:text>
7012     <xsl:value-of select="name()"/>
7013     <xsl:text></xsl:text>
7014     <xsl:value-of select="@xml:id"/>
7015     <xsl:text>&#xa;%&lt;</xsl:text>
7016     <xsl:value-of select="$thisdocid"/>
7017     <xsl:text>>\Provides</xsl:text>
7018     <xsl:value-of select="$thisdocfulltype"/>
7019     <xsl:text>{</xsl:text>
7020     <xsl:value-of select="$thisdocname"/>
7021     <xsl:text>}</xsl:text>
7022     <xsl:value-of
7023         select="translate($date, '-', '/')"/>
7024     <xsl:text> v</xsl:text>
7025     <xsl:value-of select="/db:book/@version"/>
7026     <xsl:text>.</xsl:text>
7027     <xsl:value-of select="/db:book/@revision"/>
7028     <xsl:text>&#xa;</xsl:text>
7029     <xsl:text>%&lt;</xsl:text>
7030     <xsl:value-of select="$thisdocid"/>
7031     <xsl:text>> </xsl:text>
7032     <xsl:value-of

```

```

7033         select="normalize-space(replace(
7034             db:title,'\\LaTeX\\{\\}','LaTeX'))"/>
7035         <xsl:text>]&#xa;% \fi&#xa;</xsl:text>
7036     </xsl:if>
7037 </xsl:if>
7038 <!-- end of selective chapcpk -->

```

Implement a hard-wired class-specific preload: the fix-cm package, without which all font sizes are restricted to the step-sizes of the original \LaTeX .

makechapapp

TODO: Long-term, move these to a mechanism similar to prepost

```

7039 <!-- CLASSES ONLY -->
7040 <xsl:if test="$doctype='class'
7041             and
7042             not(name()='appendix')">
7043     <xsl:text>% \begin{CPKpackage}{fix-cm}
7044 % Preloaded functions to override the default \LaTeX\
7045 % step-size font sizes (which can still be used,
7046 % but are no longer restrictions).\par
7047 % </xsl:text>
7048     <xsl:text>\begin</xsl:text>
7049     <xsl:text>{macrocode}
7050 \RequirePackage{fix-cm}
7051 % </xsl:text>
7052     <xsl:text>\end</xsl:text>
7053     <xsl:text>{macrocode}
7054 % \end{CPKpackage}&#xa;</xsl:text>
7055 </xsl:if>

```

Document first how we handle the autogeneration of `\PassOptionsToPackage` commands unless overridden manually. We should here be in the first chapter of the "code" part.

makechapapp

TODO: This should really be generated, not embedded verbatim

The only candidate is the xcolor package for which *ClassPack* uses the `svgnames` option. This needs preloading to prevent a clash with other packages (eg hyperref). There is an exclusion for packages which use themselves in their own documentation (eg for examples). The action is also echoed to the terminal when the package is used.

```

7056 <xsl:choose>
7057 <xsl:when test="@condition='noauto'">
7058     <xsl:text>% \par Auto-generation of options to
7059 % pass to packages is suppressed in this file.&#xa;</xsl:text>
7060 </xsl:when>
7061 <xsl:otherwise>
7062     <!-- special handling for [svgnames]{xcolor} -->
7063     <xsl:if
7064         test="//db:constraintdef[@xml:id='clspackages']
7065             //db:seg[.='xcolor']

```

```

7066             [contains(@role,'svgnames')
7067             or
7068             not(@role)]
7069             [not(@condition='only')]"]>
7070             <xsl:text>% \begin{CPKoption}{svgnames}
7071 % Pass the \textbf{\texttt{svgnames}} option to the
7072 % \textsf{xcolor} package if that gets loaded later.
7073 % This avoids a conflict with any other packages in the class
7074 % (eg \textsf{hyperref}) which use their own default
7075 % when they load \textsf{xcolor}.\par&#xa;</xsl:text>
7076 <!-- EXCEPTION HANDLER WHEN PACKAGE ITSELF INCLUDED removed
7077             <xsl:if test="/db:book/@xlink:role
7078             and
7079             $filetype='sty'">
7080             <xsl:text>% However, we have to make
7081 % an exception in this case because the package gets
7082 % used in its own documentation, which would cause a
7083 % duplicate \verb+\PassOptionsToPackage+, so we code
7084 % around it by testing the current package name against
7085 % the job name of the calling \verb+.dtx+ file~ if
7086 % they are the same, then this is the case in point,
7087 % and the \verb+\PassOptionsToPackage+ command is
7088 % \emph{not} executed; otherwise it is OK to include it.\par
7089 %     </xsl:text>
7090             <xsl:text>\begin</xsl:text>
7091             <xsl:text>{macrocode}
7092 \def\CPKthispackage{</xsl:text>
7093             <xsl:value-of select="$docname"/>
7094             <xsl:text>}
7095 \edef\CPKthispackage{\meaning\CPKthispackage}
7096 \edef\CPKthisjob{\jobname}
7097 \edef\CPKthisjob{\meaning\CPKthisjob}
7098 \ifx\CPKthispackage\CPKthisjob
7099 %% this is the documentation: omit PassOptionsToPackage
7100 \PackageWarning{classpack}{Option svgnames has not
7101   been passed ^^Jto package xcolor (because this is
7102   the package documentation)}
7103 \else
7104 %% this is a user job: include PassOptionsToPackage
7105 \PackageWarning{classpack}{Option svgnames is being
7106   passed ^^Jto package xcolor (to avoid later clashes)}
7107 %     </xsl:text>
7108             <xsl:text>\end</xsl:text>
7109             <xsl:text>{macrocode}&#xa;</xsl:text>
7110             </xsl:if>
7111 -->

```

Now do the actual preloading. The nested embedded conditional switch technique is due to user zeroth at <https://tex.stackexchange.com/questions/44499/how-to-test-jobname-compilation-option-within-latex-file/54895>. Note that the earlier automation of the **hyphens** option to the url package must now be a \documentclass option.

```

7112             <xsl:text>%     </xsl:text>

```

```

7113         <xsl:text>\begin</xsl:text>
7114         <xsl:text>{macrocode}
7115     \PassOptionsToPackage{svgnames}{xcolor}
7116 %     </xsl:text>
7117         <xsl:text>\end</xsl:text>
7118         <xsl:text>{macrocode}&#xa;</xsl:text>
7119         <!--
7120         <xsl:if
7121             test="/db:book/@xlink:role
7122                 and
7123                 $filetype='sty'">
7124         <xsl:text>%     </xsl:text>
7125         <xsl:text>\begin</xsl:text>
7126         <xsl:text>{macrocode}
7127 \fi
7128 %     </xsl:text>
7129         <xsl:text>\end</xsl:text>
7130         <xsl:text>{macrocode}&#xa;</xsl:text>
7131     </xsl:if>
7132 -->
7133         <xsl:text>% \end{CPKoption}&#xa;</xsl:text>
7134     </xsl:if>
7135 </xsl:otherwise>
7136 </xsl:choose>

```

Any preliminary non-`<sect1>` material was already output before the auto-initialisation, so now do the rest of the chapter/appendix. After the call to `checkpackages`, if the current document is a shell script, make sure it terminates here, because Do \TeX will output an `\endinput` and two \LaTeX comment lines which will confuse a script. This ends processing for a code chapter.

```

7137     <xsl:choose>
7138     <!-- we did anything prior to the sect1 earlier
7139         so now do the sect1 and everything following -->
7140     <xsl:when test="db:sect1">
7141     <xsl:apply-templates
7142         select="db:sect1[1] |
7143             db:sect1[1]/following-sibling::*"/>
7144     </xsl:when>
7145     <!-- if there is no sect1, nothing will have been
7146         output earlier, so do it all now -->
7147     <xsl:otherwise>
7148     <xsl:apply-templates/>
7149     </xsl:otherwise>
7150     </xsl:choose>
7151     <!-- Unclear if this is still necessary -->
7152     <xsl:call-template name="checkpackages">
7153     <xsl:with-param name="pos" select="'after'"/>
7154     <xsl:with-param name="loc" select="."/>
7155     </xsl:call-template>
7156     <!-- scripts only -->
7157     <xsl:if test="$doctype='script'">
7158     <xsl:text>% [Exit added by \textit{ClassPack}
7159 % to shield shell from \textsf{Doc\TeX}.\]\par

```

```

7160 % \iffalse&#xa;##
7161 ## Exit added by ClassPack to shield shell from DocTeX.
7162 % \fi&#xa;%    </xsl:text>
7163         <xsl:text>\begin</xsl:text>
7164         <xsl:text>{macrocode}&#xa;exit 0</xsl:text>
7165         <xsl:text>&#xa;%    </xsl:text>
7166         <xsl:text>\end</xsl:text>
7167         <xsl:text>{macrocode}&#xa;</xsl:text>
7168     </xsl:if>
7169 </xsl:when><!-- end of dealing with part/code -->

```

Other circumstances probably include the licence document, passed through untouched.

```

7170         <xsl:otherwise>
7171         <xsl:apply-templates/>
7172     </xsl:otherwise>
7173 </xsl:choose>
7174 </xsl:template>

```

checkpackages This template checks whether the current document location is the correct place for deferred packages to be included. The \$pos parameter is the "before" or "after" @role value passed from the call; the \$loc parameter is the location in the document (ID) where this call was made from.

If there is a matching class or style package <constraintdef> whose <IDREF> matches this section's <ID> and whose @role attribute matches \$pos (or if there is no @role at all), do this template.

```

7175 <xsl:template name="checkpackages">
7176   <xsl:param name="pos"/><!-- before or after -->
7177   <xsl:param name="loc"/><!-- name of calling element type -->
7178   <xsl:variable name="thisgi"
7179     select="$thisdoc/db:book/db:info/db:cover/db:constraintdef
7180       [@xml:id=concat($filetype,'packages')]/@linkend"/>
7181   <xsl:variable name="thisrole"
7182     select="$thisdoc/db:book/db:info/db:cover/db:constraintdef
7183       [@xml:id=concat($filetype,'packages')]/@role"/>
7184   <!--
7185   <xsl:text>% Checking in </xsl:text>
7186   <xsl:value-of select="name()"/>
7187   <xsl:text>/</xsl:text>
7188   <xsl:value-of select="@xml:id"/>
7189   <xsl:text> for package location </xsl:text>
7190   <xsl:value-of select="$pos"/>
7191   <xsl:text> </xsl:text>
7192   <xsl:value-of select="name()"/>
7193   <xsl:text>/</xsl:text>
7194   <xsl:value-of select="$thisgi"/>
7195   <xsl:text>&#xa;</xsl:text>
7196   -->
7197   <xsl:if
7198     test="$thisgi
7199       and

```

```

7200         current()/@xml:id
7201         and
7202         $thisgi=current()/@xml:id
7203         and
7204         ($pos=$thisrole
7205         or
7206         not($thisrole))">
7207     <xsl:variable name="gi" select="local-name()"/>
7208     <xsl:text>%\iffalse
7209     %%&#xa;</xsl:text>
7210     <!--
7211     <xsl:text>%% [in checkpackages with pos=</xsl:text>
7212     <xsl:value-of select="$pos"/>
7213     <xsl:text>, loc=</xsl:text>
7214     <xsl:value-of select="substring($loc,1,32)"/>
7215     <xsl:text>, thisgi=</xsl:text>
7216     <xsl:value-of select="$thisgi"/>
7217     <xsl:text>, thisrole=</xsl:text>
7218     <xsl:value-of select="$thisrole"/>
7219     <xsl:text>&#xa;</xsl:text>
7220     -->
7221     <xsl:text>%% Packages required for the </xsl:text>
7222     <xsl:value-of select="$doctype"/>
7223     <xsl:text>
7224     %%
7225     % \fi
7226     % \</xsl:text>
7227     <xsl:choose>
7228         <!-- $gi is the element type nominated by the @linkend
7229             attribute in the stypackage or clspackages
7230             constraintdef where the class/package packages
7231             are declared -->
7232         <xsl:when test="$gi='chapter' or $gi='appendix'">
7233             <xsl:text>clearpage&#xa;% \section</xsl:text>
7234         </xsl:when>
7235         <xsl:when test="$gi='sect1'">
7236             <xsl:text>subsection</xsl:text>
7237         </xsl:when>
7238         <xsl:when test="$gi='sect2'">
7239             <xsl:text>subsubsection</xsl:text>
7240         </xsl:when>
7241         <xsl:when test="$gi='sect3'">
7242             <xsl:text>paragraph</xsl:text>
7243         </xsl:when>
7244         <xsl:when test="$gi='sect4'">
7245             <xsl:text>subparagraph</xsl:text>
7246         </xsl:when>
7247         <xsl:otherwise>
7248             <xsl:message>
7249                 <xsl:text>WARNING: </xsl:text>
7250                 <xsl:value-of select="$gi"/>
7251                 <xsl:text> is an unidentifiable </xsl:text>
7252                 <xsl:text>place to declare packages</xsl:text>
7253             </xsl:message>

```

```

7254         <xsl:text>\PackageWarning{classpack}{Unidentifiable place `</xsl:text>
7255         <xsl:value-of select="$thisgi"/>
7256         <xsl:text>' to declare packages}</xsl:text>
7257     </xsl:otherwise>
7258 </xsl:choose>
7259 <xsl:text>{</xsl:text>
7260 <xsl:choose>
7261     <xsl:when
7262         test="/db:book/db:info/db:cover
7263             /db:constraintdef
7264             [@linkend=current()/@xml:id]
7265             /db:segmentedlist/db:segtitle">
7266     <xsl:apply-templates
7267         select="/db:book/db:info/db:cover
7268             /db:constraintdef
7269             [@linkend=current()/@xml:id]
7270             /db:segmentedlist/db:segtitle/node()"/>
7271     </xsl:when>
7272     <xsl:otherwise>
7273         <xsl:text>Packages loaded</xsl:text>
7274     </xsl:otherwise>
7275 </xsl:choose>
7276 <xsl:text>}}\label{</xsl:text>
7277 <xsl:value-of
7278     select="/db:book/db:info/db:cover
7279         /db:constraintdef
7280         [@linkend=current()/@xml:id]
7281         /@xml:id"/>
7282 <xsl:text>}}&#xa;</xsl:text>
7283 <!-- INCLUDE COMPULSORY PACKAGE[s] for cls/sty -->
7284 <xsl:if test="($filetype='cls' or $filetype='sty')
7285     and
7286     count($prepost/db:procedure[@xml:id='prepackage'
7287         /db:step[contains(@condition,$filetype)]
7288         [db:constructorsynopsis/@condition='']) > 0">
7289     <xsl:text>% Packages required for operation:&#xa;</xsl:text>
7290     <xsl:for-each
7291         select="$prepost/db:procedure[@xml:id='prepackage'
7292             /db:step[contains(@condition,$filetype)]
7293             [db:constructorsynopsis/@condition='']">
7294         <xsl:text>% \begin{CPKpackage}{</xsl:text>
7295         <xsl:value-of select="@remap"/>
7296         <xsl:text>}}&#xa;</xsl:text>
7297         <xsl:apply-templates select="db:para"/>
7298         <xsl:text>% \iffalse&#xa;%</xsl:text>
7299         <xsl:value-of select="normalize-space(db:para[1])"/>
7300         <xsl:text>&#xa;% \fi&#xa;</xsl:text>
7301         <xsl:text>% \begin{macrocode}&#xa;</xsl:text>
7302         <xsl:text>\RequirePackage{</xsl:text>
7303         <xsl:value-of select="@remap"/>
7304         <xsl:text>}}&#xa;</xsl:text>
7305     <xsl:if
7306         test="db:constraintdef[db:cmdsynopsis[db:command]]">
7307     <xsl:for-each

```

```

7308         select="db:constraintdef/db:cmdsynopsis/db:command">
7309         <xsl:value-of select="."/>
7310         <xsl:text>&#xa;</xsl:text>
7311     </xsl:for-each>
7312 </xsl:if>
7313 <xsl:text>% \end</xsl:text>
7314 <xsl:text>{macrocode}&#xa;</xsl:text>
7315 <!-- and add on any code from the postpackage section -->
7316 <xsl:for-each
7317     select="$prepost/db:procedure[@xml:id='postpackage']
7318           /db:step[contains(@condition,$filetype)]
7319           [@remap=current()/@remap]">
7320     <xsl:text>% </xsl:text>
7321     <xsl:value-of select="normalize-space(db:para[1])"/>
7322     <xsl:text>&#xa;</xsl:text>
7323     <xsl:text>% \begin{macrocode}&#xa;</xsl:text>
7324     <xsl:if
7325         test="db:constraintdef[db:cmdsynopsis[db:command]]">
7326         <xsl:for-each
7327             select="db:constraintdef/db:cmdsynopsis/db:command">
7328             <xsl:value-of select="."/>
7329             <xsl:text>&#xa;</xsl:text>
7330         </xsl:for-each>
7331         </xsl:if>
7332         <xsl:text>% \end</xsl:text>
7333         <xsl:text>{macrocode}&#xa;</xsl:text>
7334     </xsl:for-each>
7335     <xsl:text>% \end{CPKpackage}&#xa;</xsl:text>
7336 </xsl:for-each>
7337 </xsl:if><!-- end of required packages for cls/sty -->
7338 <!-- now each requested package -->
7339 <xsl:for-each
7340     select="/db:book/db:info/db:cover
7341           /db:constraintdef
7342           [@linkend=current()/@xml:id]
7343           /db:segmentedlist/db:seglistitem
7344           [db:seg!='']">
7345     <xsl:call-template name="packages">
7346         <!-- context is seglistitem from cls/sty packages -->
7347         <xsl:with-param name="pkg" select="."/>
7348         <xsl:with-param name="dest" select="$filetype"/>
7349         <xsl:with-param name="cmdblocks"
7350             select="$prepost/db:procedure[@xml:id='postpackage']/
7351                   db:step[@remap=current()/db:seg]/
7352                   db:constraintdef/db:cmdsynopsis/
7353                   db:command/@conformance"/>
7354         <xsl:with-param name="mode" select="''deferred''"/>
7355         <xsl:with-param name="loc" select="$loc"/>
7356     </xsl:call-template>
7357 </xsl:for-each>
7358 <xsl:text>% &#xa;</xsl:text>
7359 <xsl:if
7360     test="/db:book/db:info/db:cover
7361           /db:constraintdef

```

```

7362         [@linkend=current()/@xml:id]
7363         /@xml:id!='stypackages'
7364         and
7365         /db:book/db:info/db:cover
7366         /db:constraintdef
7367         [@linkend=current()/@xml:id]
7368         /@xml:id!='clspackages'"]>
7369     <xsl:message>
7370       <xsl:text>WARNING: packages listed in "</xsl:text>
7371       <xsl:value-of select="$gi"/>
7372       <xsl:text>" that doesn't belong to this </xsl:text>
7373       <xsl:value-of select="$doctype"/>
7374       <xsl:text> (xml:id="</xsl:text>
7375       <xsl:value-of select="@xml:id"/>
7376       <xsl:text>").</xsl:text>
7377     </xsl:message>
7378     <xsl:text>% WARNING: packages listed in a \verb`</xsl:text>
7379     <xsl:value-of select="$gi"/>
7380     <xsl:text>` element of the master document </xsl:text>
7381     <xsl:text>that was not flagged as belonging to this </xsl:text>
7382     <xsl:value-of select="$doctype"/>
7383     <xsl:text> (\verb`xml:id="</xsl:text>
7384     <xsl:value-of select="@xml:id"/>
7385     <xsl:text>`)&#xa;</xsl:text>
7386   </xsl:if>
7387 </xsl:if>
7388 </xsl:template>

```

11.5.3 External files

`readme` This template creates the `README.md` file from the boilerplate in `readme.xml`. This uses the `readme` mode and the special indirections described in [section 11.5.3](#) for inserting values from the user's *ClassPack* document into the `README.md` file.

```

7389 <xsl:template name="readme">
7390   <xsl:variable name="file">
7391     <xsl:text>README.md</xsl:text>
7392   </xsl:variable>
7393   <xsl:result-document format="textFormat"
7394     href="{ $file }">
7395     <xsl:text>&#xa;[//]: # (the </xsl:text>
7396     <xsl:value-of select="$docname"/>
7397     <xsl:text> LaTeX </xsl:text>
7398     <xsl:value-of select="$doctype"/>
7399     <xsl:text>)&#xa;&#xa;# The </xsl:text>
7400     <xsl:value-of select="$docname"/>
7401     <xsl:text> LaTeX2ε package&#xa;&#xa;This is the </xsl:text>
7402     <xsl:value-of select="$file"/>
7403     <xsl:text> file for the </xsl:text>
7404     <xsl:value-of select="$docname"/>
7405     <xsl:text> LaTeX </xsl:text>
7406     <xsl:value-of select="$doctype"/>
7407     <xsl:text> v</xsl:text>
7408     <xsl:value-of select="$latestrevhist"/>

```

```

7409     <xsl:text> (</xsl:text>
7410     <xsl:value-of select="$date"/>
7411     <xsl:text>)&#xa;generated by ClassPack v.</xsl:text>
7412     <xsl:value-of select="$thisversion"/>
7413     <xsl:text> (</xsl:text>
7414     <xsl:value-of select="$thisverdate"/>
7415     <xsl:text>) on </xsl:text>
7416     <xsl:value-of
7417       select="format-dateTime(current-dateTime(),
7418         '[D] [MNn] [Y] at [H]:[m]:[s] ')/>
7419     <xsl:text>&#xa;</xsl:text>
7420     <xsl:apply-templates select="$readme/*"
7421       mode="readme"/>
7422     <xsl:if test="$readme//db:footnote">
7423       <xsl:text>---&#xa;#### Footnotes&#xa;</xsl:text>
7424       <xsl:for-each select="$readme//db:footnote">
7425         <xsl:number format="1." count="db:footnote"
7426           level="any"/>
7427         <xsl:apply-templates select="node()"
7428           mode="inline"/>
7429         <xsl:text>&#xa;</xsl:text>
7430       </xsl:for-each>
7431     </xsl:if>
7432   </xsl:result-document>
7433 </xsl:template>

```

manifest In a similar manner to the README.md, this template creates the MANIFEST file, but has no formatting requirements, only listed content. Three files are always present: README.md, MANIFEST, and VERSION. The TDS file, which was previously embedded in the non-TDS distribution, is no longer included. The remaining files are included as needed: .dtx, .ins, .pdf, any ancillary files generated, and any additional files listed manually by the author; but *not* the class or style file itself: that is generated by the user extracting it from the .dtx file.

manifest **TODO: Add generated secondary files, and automate images from the doc**

```

7434   <xsl:template name="manifest">
7435     <xsl:variable name="file">
7436       <xsl:text>MANIFEST</xsl:text>
7437     </xsl:variable>
7438     <xsl:result-document format="textFormat"
7439       href="{ $file }">
7440       <!-- THESE THREE ALWAYS -->
7441       <xsl:text>README.md
7442       MANIFEST
7443       VERSION&#xa;</xsl:text>
7444       <!-- always the PDF, DTX, and INS -->
7445       <xsl:value-of select="$docname"/>
7446       <xsl:text>.dtx&#xa;</xsl:text>
7447       <xsl:value-of select="$docname"/>
7448       <xsl:text>.ins&#xa;</xsl:text>
7449       <xsl:value-of select="$docname"/>

```

```

7450     <xsl:text>.pdf&#xa;</xsl:text>
7451     <!-- the style or class file, obv -->
7452     <xsl:value-of select="$docname"/>
7453     <xsl:text>.</xsl:text>
7454     <xsl:value-of select="$filetype"/>
7455     <xsl:text>&#xa;</xsl:text>
7456     <!-- annotated appendix files and standalone
7457           unannotated files (the Makefile
7458           omits scripts from packaging )-->
7459     <xsl:for-each
7460       select="/db:book/db:part[@xml:id='code']
7461             /db:appendix[@xml:id
7462             and @xlink:href
7463             and descendant::db:programlisting
7464             and not(matches(@xlink:href,'\.cls$'))
7465             and not(matches(@xlink:href,'\.sty$'))]
7466             |
7467             /db:book/db:part[@xml:id='files']
7468             /db:chapter[@xml:id and @xlink:href
7469             and db:programlisting]">
7470       <xsl:value-of select="@xlink:href"/>
7471       <xsl:text>&#xa;</xsl:text>
7472     </xsl:for-each>
7473     <!-- IMAGE FOR DOC, should be in cover/artwork -->
7474     <xsl:if
7475       test="/db:book/db:info/db:abstract/@xlink:href
7476             and
7477             not(/db:book/db:info/db:abstract/@xlink:show='none')">
7478       <xsl:value-of
7479         select="/db:book/db:info/db:abstract/@xlink:href"/>
7480       <xsl:text>&#xa;</xsl:text>
7481     </xsl:if>
7482     <!-- ANY SPECIAL REQUESTS -->
7483     <xsl:for-each
7484       select="/db:book/db:info/db:cover
7485             /db:constraintdef[@xml:id='manifest']
7486             /db:simplelist/db:member[.='']">
7487       <xsl:value-of select="normalize-space(.)"/>
7488       <xsl:text>&#xa;</xsl:text>
7489     </xsl:for-each>
7490   </xsl:result-document>
7491 </xsl:template>

```

11.5.4 Utilities

`makecommentchar` Set the comment character to be the one whose name matches the `@arch` attribute of the closest chapter, appendix, or book (root) ancestor and whose `@filetype` attribute matches the `@userlevel` attribute of the same ancestor, all using the `$commentchars` variable.

```

7492   <xsl:template name="makecommentchar">
7493     <xsl:variable name="closest"
7494       select="(ancestor::db:book |
7495             ancestor::db:chapter |

```

```

7496         ancestor::db:appendix)[1]"/>
7497     <xsl:value-of
7498         select="$commentchars/*
7499             [name()=$closest/@arch]
7500             [@filetype=$closest/@userlevel]"/>
7501 </xsl:template>

```

`makesentinel` This does the same as `makecommentchar` but excludes the terminator for non-XML file types.

```

7502 <xsl:template name="makesentinel">
7503     <xsl:variable name="closest"
7504         select="(ancestor::db:book |
7505             ancestor::db:chapter |
7506             ancestor::db:appendix)[1]"/>
7507     <xsl:variable name="filetype"
7508         select="$commentchars/*
7509             [name()=$closest/@arch]
7510             [@filetype=$closest/@userlevel]"/>
7511     <xsl:value-of select="$filetype"/>
7512     <!-- repeat for non-XML file types -->
7513     <xsl:if test="$filetype[not(@term)]">
7514         <xsl:value-of select="$filetype"/>
7515     </xsl:if>
7516     <xsl:text> </xsl:text>
7517 </xsl:template>

```

`maketermsentinel` This template creates the terminator for `makesentinel`.

```

7518 <xsl:template name="maketermsentinel">
7519     <xsl:variable name="closest"
7520         select="(ancestor::db:book |
7521             ancestor::db:chapter |
7522             ancestor::db:appendix)[1]"/>
7523     <xsl:variable name="filetype"
7524         select="$commentchars/*
7525             [name()=$closest/@arch]
7526             [@filetype=$closest/@userlevel]"/>
7527     <xsl:text> </xsl:text>
7528     <xsl:value-of select="$filetype/@term"/>
7529     <!-- repeat for non-XML file types -->
7530     <xsl:if test="$filetype[not(@term)]">
7531         <xsl:value-of select="$filetype/@term"/>
7532     </xsl:if>
7533 </xsl:template>

```

`makefence` Make a fence (line of repeated characters) from the appropriate *local* `$sentinel` character and its terminator by repeating it 66 times.

```

7534 <xsl:template name="makefence">
7535     <xsl:variable name="sentinel">
7536         <xsl:call-template name="makesentinel"/>
7537     </xsl:variable>

```

```

7538     <xsl:variable name="termsentinel">
7539       <xsl:call-template name="maketermsentinel"/>
7540     </xsl:variable>
7541     <xsl:value-of select="$sentinel"/>
7542     <xsl:for-each select="for $i in 1 to 66 return $i">
7543       <xsl:text>*</xsl:text>
7544     </xsl:for-each>
7545     <xsl:value-of select="$termsentinel"/>
7546     <xsl:text>&#xa;</xsl:text>
7547   </xsl:template>

```

starttag Output a DocTeX start-tag which delimits extractable text, either using the ID of the document, or some other value passed as the parameter *\$identity*.

```

7548   <xsl:template name="start-tag">
7549     <xsl:param name="identity">
7550       <xsl:value-of select="@xml:id"/>
7551     </xsl:param>
7552     <xsl:text>% \iffalse&#xa;%&lt;*</xsl:text>
7553     <xsl:value-of select="$identity"/>
7554     <xsl:text>>&#xa;% \fi&#xa;</xsl:text>
7555   </xsl:template>

```

endtag Output a DocTeX end-tag on the same basis as the start-tag.

```

7556   <xsl:template name="end-tag">
7557     <xsl:param name="identity">
7558       <xsl:value-of select="@xml:id"/>
7559     </xsl:param>
7560     <xsl:text>% \iffalse&#xa;%&lt;*</xsl:text>
7561     <xsl:value-of select="$identity"/>
7562     <xsl:text>>&#xa;% \fi&#xa;</xsl:text>
7563   </xsl:template>

```

silcommentchar This template and its related terminator and sentinel are used where the standard comment character, terminator, and sentinel are already in use, and an ancillary document is being output (eg in an appendix) which requires a different syntax.

```

7564   <xsl:function name="sil:commentchar">
7565     <xsl:param name="context"/>
7566     <xsl:variable name="closest"
7567       select="($context/ancestor-or-self::db:book[1] |
7568         $context/ancestor-or-self::db:chapter[1] |
7569         $context/ancestor-or-self::db:appendix[1])[last()]">
7570     <xsl:variable name="thisdoctype"
7571       select="tokenize($closest/@xlink:href,'\.')
7572         [last()]">
7573     <xsl:value-of
7574       select="($commentchars/*[name()=$closest/@arch][1] |
7575         $commentchars/*[@filetype=$closest/@userlevel][1] |
7576         $commentchars/*[@filetype=$thisdoctype][1])[last()]">
7577   </xsl:function>

```

`siltermcommentchar` As above

```

7578 <xsl:function name="sil:termcommentchar">
7579   <xsl:param name="context"/>
7580   <!-- no parameters: evaluates using current context -->
7581   <xsl:variable name="closest"
7582     select="$context/ancestor-or-self::*
7583       [name()='book'
7584         or
7585         name()='chapter'
7586         or
7587         name()='appendix'] [1]"/>
7588   <xsl:variable name="filetype"
7589     select="$commentchars/*
7590       [name()=$closest/@arch]
7591       [@filetype=$closest/@userlevel] [1]"/>
7592   <xsl:value-of select="$filetype/@term"/>
7593 </xsl:function>

```

`silsentinel` As above

```

7594 <xsl:function name="sil:sentinel">
7595   <xsl:param name="context"/>
7596   <xsl:value-of select="sil:commentchar($context)"/>
7597   <xsl:if test="sil:commentchar($context)!='&lt;!--'">
7598     <xsl:value-of select="sil:commentchar($context)"/>
7599   </xsl:if>
7600 </xsl:function>

```

`avoidverb` Try to output monospace filenames, URIs, etc while avoiding the use of `\verb`. Start by picking a delimiter to use if we really do have to use `\verb`. In each case, a prefix and suffix may be added depending on the language the token belongs to.

```

7601 <xsl:template name="avoidverb">
7602   <xsl:variable name="delim">
7603     <xsl:choose>
7604       <xsl:when test="not(contains(.,'|'))">
7605         <xsl:text>|</xsl:text>
7606       </xsl:when>
7607       <xsl:when test="not(contains(.,'+'))">
7608         <xsl:text>+</xsl:text>
7609       </xsl:when>
7610       <xsl:otherwise>
7611         <xsl:text>`</xsl:text>
7612       </xsl:otherwise>
7613     </xsl:choose>
7614   </xsl:variable>

```

`avoidverb` **Verbatim with T_EX special characters**

If the content contains any of T_EX's special characters or the visible space
 (`\ $ ^ { } < | > _`)

that are also invalid or uncomfortable in URIs, we will have to use `\verb`.

```

7615     <xsl:choose>
7616         <xsl:when test="matches(.,'.*[\$^{}&lt;\&gt;_].*')
7617             and not(ancestor::db:title)
7618             and not(ancestor::db:footnote)
7619             and not(ancestor::db:term)"/>

```

If the content contains a `<replaceable>` element or has the specific exclusion condition `"nolit"`, it will need to be split into abutting fragments by using `<xsl:apply-templates>`.

```

7620     <xsl:choose>
7621         <xsl:when
7622             test="db:replaceable or @condition='nolit'">
7623             <xsl:text>\verb</xsl:text>
7624             <xsl:value-of select="$delim"/>
7625             <!-- PREFIXING -->
7626             <xsl:if test="(name()='command'
7627                 or
7628                 @role='length')
7629                 and
7630                 (@language='TeX'
7631                 or
7632                 @language='LaTeXe'
7633                 or
7634                 @language='LaTeX'
7635                 or
7636                 not(@language))">
7637                 <xsl:text>\</xsl:text>
7638             </xsl:if>
7639             <xsl:if test="@language='XSLT'
7640                 and
7641                 (name()='varname' and
7642                 (@role='variable' or
7643                 @role='parameter'))
7644                 or
7645                 name()='parameter'">
7646                 <xsl:text>$</xsl:text>
7647             </xsl:if>
7648             <xsl:if test="name()='varname'
7649                 and
7650                 @language='Makefile'
7651                 and
7652                 @role='variable'">
7653                 <xsl:text>$(</xsl:text>
7654             </xsl:if>
7655             <xsl:if test="name()='varname'
7656                 and
7657                 @language='BiBTeX'
7658                 and
7659                 @role='entrytype'">
7660                 <xsl:text>@</xsl:text>
7661             </xsl:if>

```

```

7662         <xsl:apply-templates/>
7663         <!-- SUFFIXING -->
7664         <xsl:if test="@xlink:type">
7665             <xsl:text>.</xsl:text>
7666             <xsl:value-of select="@xlink:type"/>
7667         </xsl:if>
7668         <xsl:if test="name()='varname'
7669                     and
7670                     @language='Makefile'
7671                     and
7672                     @role='variable'">
7673             <xsl:text>></xsl:text>
7674         </xsl:if>
7675         <xsl:value-of select="$delim"/>
7676     </xsl:when>

```

Otherwise just the plain `\verb` is used with `normalize-space` of the content.

```

7677     <xsl:otherwise>
7678         <xsl:text>\verb</xsl:text>
7679         <xsl:value-of select="$delim"/>
7680         <xsl:if test="(name()='command'
7681                     or
7682                     @role='length')
7683                     and
7684                     (@language='TeX'
7685                     or
7686                     @language='LaTeXe'
7687                     or
7688                     @language='LaTeX'
7689                     or
7690                     not(@language))">
7691             <xsl:text>\</xsl:text>
7692         </xsl:if>
7693         <xsl:if test="@language='XSLT'
7694                     and
7695                     (name()='varname' and
7696                     (@role='variable' or
7697                     @role='parameter'))
7698                     or
7699                     name()='parameter'">
7700             <xsl:text>${</xsl:text>
7701         </xsl:if>
7702         <xsl:if test="name()='varname'
7703                     and
7704                     @language='Makefile'
7705                     and
7706                     @role='variable'">
7707             <xsl:text>${</xsl:text>
7708         </xsl:if>
7709         <xsl:if test="name()='varname'
7710                     and
7711                     @language='BiBTeX'
7712                     and

```

```

7713             @role='entrytype'">
7714         <xsl:text>@</xsl:text>
7715     </xsl:if>
7716     <xsl:value-of select="normalize-space(.)"/>
7717     <xsl:if test="@xlink:type">
7718         <xsl:text>.</xsl:text>
7719         <xsl:value-of select="@xlink:type"/>
7720     </xsl:if>
7721     <xsl:if test="name()='varname'
7722             and
7723             @language='Makefile'
7724             and
7725             @role='variable'">
7726         <xsl:text>></xsl:text>
7727     </xsl:if>
7728     <xsl:value-of select="$delim"/>
7729 </xsl:otherwise>
7730 </xsl:choose>
7731 </xsl:when>

```

avoidverb **Verbatim in titles**

The use of verbatim text in titles, footnotes, and the terms of variable lists is not yet fully tested. so it is implemented with the `\url` command.

avoidverb **TODO: Needs more testing**

```

7732     <xsl:when test="matches(.,'.*[\$^{}&lt;\&gt;].*')
7733             and
7734             (ancestor::db:title or
7735             ancestor::db:footnote or
7736             ancestor::db:term)">
7737     <xsl:message>
7738         <xsl:text>Warning: verbatim in </xsl:text>
7739         <xsl:value-of select="name(ancestor::*[3])"/>
7740         <xsl:text></xsl:text>
7741         <xsl:value-of select="name(ancestor::*[2])"/>
7742         <xsl:text></xsl:text>
7743         <xsl:value-of select="name(ancestor::*[1])"/>
7744         <xsl:text> for "</xsl:text>
7745         <xsl:value-of select="normalize-space(.)"/>
7746         <xsl:text>"</xsl:text>
7747     </xsl:message>
7748     <xsl:text>\url{</xsl:text>
7749     <xsl:value-of select="normalize-space(.)"/>
7750     <xsl:text>}</xsl:text>
7751 </xsl:when>

```

avoidverb **Basic monospace for non-special content**

Test for the presence of URI specials (`& _ ~ # %`).

```

7752     <xsl:otherwise>
7753     <xsl:choose>

```

```

7754      <xsl:when
7755          test="matches(.,'.*[\&_~%].*')
7756              and
7757              not(db:replaceable or @condition='nolit')">
7758          <xsl:text>\url{</xsl:text>
7759          <xsl:value-of select="normalize-space(.)"/>
7760          <xsl:text>}</xsl:text>
7761      </xsl:when>

```

The same test for use of <replaceable> is needed as above.

```

7762      <xsl:when
7763          test="db:replaceable or @condition='nolit'">
7764          <xsl:text>\ttfamily{</xsl:text>
7765          <xsl:if
7766              test="(name()='command'
7767                  or
7768                  @role='length')
7769                  and
7770                  (@language='TeX'
7771                  or
7772                  @language='LaTeXe'
7773                  or
7774                  @language='LaTeX'
7775                  or
7776                  not(@language))">
7777              <xsl:text>\textbackslash{</xsl:text>
7778          </xsl:if>
7779          <xsl:if test="@language='XSLT'
7780                  and
7781                  (name()='varname' and
7782                  (@role='variable' or
7783                  @role='parameter'))
7784                  or
7785                  name()='parameter'">
7786              <xsl:text>\${</xsl:text>
7787          </xsl:if>
7788          <xsl:if test="name()='varname'
7789                  and
7790                  @language='Makefile'
7791                  and
7792                  @role='variable'">
7793              <xsl:text>\${</xsl:text>
7794          </xsl:if>
7795          <xsl:if test="name()='varname'
7796                  and
7797                  @language='BiBTeX'
7798                  and
7799                  @role='entrytype'">
7800              <xsl:text>@</xsl:text>
7801          </xsl:if>
7802          <xsl:apply-templates/>
7803          <xsl:if test="@xlink:type">
7804              <xsl:text>.</xsl:text>

```

```

7805         <xsl:value-of select="@xlink:type"/>
7806     </xsl:if>
7807     <xsl:if test="name()='varname'
7808                 and
7809                 @language='Makefile'
7810                 and
7811                 @role='variable'">
7812         <xsl:text></xsl:text>
7813     </xsl:if>
7814     <xsl:text></xsl:text>
7815 </xsl:when>

```

Failing all else, it's just plain monospace.

```

7816     <xsl:otherwise>
7817     <xsl:text>{\ttfamily{}</xsl:text>
7818     <xsl:if
7819         test="(name()='command'
7820               or
7821               @role='length')
7822               and
7823               (@language='TeX'
7824               or
7825               @language='LaTeXe'
7826               or
7827               @language='LaTeX'
7828               or
7829               not(@language))">
7830         <xsl:text>\textbackslash{}</xsl:text>
7831     </xsl:if>
7832     <xsl:if test="@language='XSLT'
7833                 and
7834                 (name()='varname' and
7835                 (@role='variable' or
7836                 @role='parameter'))
7837                 or
7838                 name()='parameter'">
7839         <xsl:text>\$</xsl:text>
7840     </xsl:if>
7841     <xsl:if test="name()='varname'
7842                 and
7843                 @language='bash'
7844                 and
7845                 @role='variable'">
7846         <xsl:text>\$</xsl:text>
7847     </xsl:if>
7848     <xsl:if test="name()='varname'
7849                 and
7850                 @language='Makefile'
7851                 and
7852                 @role='variable'">
7853         <xsl:text>\$</xsl:text>
7854     </xsl:if>
7855     <xsl:if test="name()='varname'

```

```

7856             and
7857             @language='BiBTeX'
7858             and
7859             @role='entrytype'"/>
7860         <xsl:text>@</xsl:text>
7861     </xsl:if>
7862     <xsl:value-of select="normalize-space(.)"/>
7863     <xsl:if test="@xlink:type">
7864         <xsl:text>.</xsl:text>
7865         <xsl:value-of select="@xlink:type"/>
7866     </xsl:if>
7867     <xsl:if test="name()='varname'
7868             and
7869             @language='Makefile'
7870             and
7871             @role='variable'"/>
7872         <xsl:text></xsl:text>
7873     </xsl:if>
7874     <xsl:text></xsl:text>
7875 </xsl:otherwise>
7876 </xsl:choose>
7877 </xsl:otherwise>
7878 </xsl:choose>

```

avoidverb Indexed entries

Indexing has been tested for syntax, but has not been implemented for semantics.

```

7879     <!-- finally, does it need an index entry?
7880     <xsl:variable name="gi" select="local-name()"/>
7881     <xsl:if test="tokenize(/db:book/@xreflabel,':')[.=$gi]
7882             and not(ancestor::db:warning)"/>
7883         <xsl:text>\index{</xsl:text>
7884         <xsl:value-of select="replace(.,'_','\\_')"/>
7885         <xsl:text>@<textbf{</xsl:text>
7886         <xsl:value-of select="replace(.,'_','\\_')"/>
7887         <xsl:text>}}</xsl:text>
7888     </xsl:if>
7889     -->
7890 </xsl:template>

```

htrim Perform horizontal trimming of a span of text passed in the parameter. Unlike normalization, this only replaces multiple spaces and TAB characters with a single space, and leaves leading and trailing spaces (and other types of white-space) alone.

```

7891 <xsl:template name="htrim">
7892     <xsl:param name="text"/>
7893     <xsl:variable name="TEXT"
7894         select="translate($text,'&#x9;',' ')/>
7895     <xsl:choose>
7896         <xsl:when test="contains($TEXT,' ')">
7897             <xsl:call-template name="htrim">
7898                 <xsl:with-param name="text">

```

```

7899         <xsl:value-of
7900             select="substring-before($TEXT,' ')" />
7901         <xsl:text> </xsl:text>
7902         <xsl:value-of
7903             select="substring-after($TEXT,' ')" />
7904         </xsl:with-param>
7905     </xsl:call-template>
7906 </xsl:when>
7907 <xsl:when test="contains($TEXT,'&#xa; ')">
7908     <xsl:call-template name="htrim">
7909         <xsl:with-param name="text">
7910             <xsl:value-of
7911                 select="substring-before($TEXT,'&#xa; ')" />
7912             <xsl:text>&#xa;</xsl:text>
7913             <xsl:value-of
7914                 select="substring-after($TEXT,'&#xa; ')" />
7915             </xsl:with-param>
7916         </xsl:call-template>
7917     </xsl:when>
7918     <xsl:otherwise>
7919         <xsl:value-of select="$TEXT" />
7920     </xsl:otherwise>
7921 </xsl:choose>
7922 </xsl:template>

```

`lrtrim` Left/right trimming does the reverse: it removes leading and trailing spaces but preserves embedded space. As this may be called on text within text, we pass an `addindent` `$indent` parameter which triggers a call to `addindent`.

Text in `<programlisting>` elements almost always begins with a newline, the one that comes right after the start-tag, and equally often has a newline and TAB or space character after the last printable character.

After treatment, however, we need to paste on any indent passed by the calling routine, usually to avoid trouble with documentation that references the commands `\end{lstlisting}` or `\end{verbatim}` *within* the text of the verbatim block.

```

7923 <xsl:template name="lrtrim">
7924     <xsl:param name="text" />
7925     <xsl:param name="indent" />
7926     <xsl:variable name="extent"
7927         select="string-length($text)" />
7928     <xsl:choose>
7929         <xsl:when test="substring($text,1,1)='&#xa;'">
7930             <xsl:call-template name="lrtrim">
7931                 <xsl:with-param name="text"
7932                     select="substring($text,2)" />
7933                 <xsl:with-param name="indent"
7934                     select="$indent" />
7935             </xsl:call-template>
7936         </xsl:when>
7937         <xsl:when test="substring($text,$extent)=' ' or

```

```

7938             substring($text,$extent)='&#xa;' or
7939             substring($text,$extent)='&#x9;'">
7940         <xsl:call-template name="ltrim">
7941             <xsl:with-param name="text"
7942                 select="substring($text,1,($extent - 1))"/>
7943             <xsl:with-param name="indent" select="$indent"/>
7944         </xsl:call-template>
7945     </xsl:when>
7946     <xsl:when test="$indent!=''">
7947         <xsl:call-template name="addindent">
7948             <xsl:with-param name="text" select="$text"/>
7949             <xsl:with-param name="indent" select="$indent"/>
7950         </xsl:call-template>
7951     </xsl:when>
7952     <xsl:otherwise>
7953         <xsl:value-of select="$text"/>
7954     </xsl:otherwise>
7955 </xsl:choose>
7956 </xsl:template>

```

addindent Adds the indent (spaces) given in `$indent` before every newline in `$text`.

```

7957 <xsl:template name="addindent">
7958     <xsl:param name="text"/>
7959     <xsl:param name="indent"/>
7960     <xsl:value-of select="$indent"/>
7961     <xsl:choose>
7962         <xsl:when test="contains($text,'&#xa;')">
7963             <xsl:value-of
7964                 select="substring-before($text,'&#xa;')"/>
7965             <xsl:text>&#xa;</xsl:text>
7966             <xsl:call-template name="addindent">
7967                 <xsl:with-param name="text"
7968                     select="substring-after($text,'&#xa;')"/>
7969                 <xsl:with-param name="indent"
7970                     select="$indent"/>
7971             </xsl:call-template>
7972         </xsl:when>
7973         <xsl:otherwise>
7974             <xsl:value-of select="$text"/>
7975         </xsl:otherwise>
7976     </xsl:choose>
7977 </xsl:template>

```

breakline Perform line-breaking at space tokens to bring a chunk of text to within `$max`
maxcodelen spaces wide (default is the value of `$maxcodelen`).

```

7978 <xsl:template name="breakline">
7979     <xsl:param name="string"/>
7980     <xsl:param name="max">
7981         <xsl:value-of select="$maxcodelen"/>
7982     </xsl:param>
7983     <xsl:choose>

```

```

7984 <xsl:when test="string-length($string)>$max">
7985   <!-- see if there is a substring that will fit -->
7986   <xsl:choose>
7987     <!-- if no spaces, we're stuck, so use as-is -->
7988     <xsl:when test="not(contains($string,' '))">
7989       <xsl:value-of select="$string"/>
7990     </xsl:when>
7991     <!-- if first substring fits, output/repeat -->
7992     <xsl:when
7993       test="string-length(substring-before($string,
7994         ' '))&lt;=$max">
7995       <xsl:value-of select="$max"/>
7996       <xsl:text></xsl:text>
7997       <xsl:value-of
7998         select="string-length(substring-before($string,
7999           ' '))"/>
8000       <xsl:value-of
8001         select="substring-before($string,' ')/>
8002       <xsl:text> </xsl:text>
8003       <!-- see how much we have left -->
8004       <xsl:choose>
8005         <xsl:when
8006           test="($max - 1 -
8007             string-length(substring-before($string,
8008               ' ')))
8009             &lt;
8010             string-length(substring-before(
8011               substring-after($string,
8012                 ' '), ' '))">
8013           <xsl:text>&#xa; </xsl:text>
8014           <xsl:call-template name="breakline">
8015             <xsl:with-param name="string"
8016               select="substring-after($string,' ')/>
8017           </xsl:call-template>
8018         </xsl:when>
8019         <xsl:otherwise>
8020           <xsl:call-template name="breakline">
8021             <xsl:with-param name="string"
8022               select="substring-after($string,' ')/>
8023             <xsl:with-param name="max"
8024               select="$max - 1 -
8025                 string-length(substring-before(
8026                   $string,' '))"/>
8027           </xsl:call-template>
8028         </xsl:otherwise>
8029       </xsl:choose>
8030     </xsl:when>
8031   </xsl:choose>
8032 </xsl:when>
8033 <xsl:otherwise>
8034   <xsl:value-of select="$string"/>
8035 </xsl:otherwise>
8036 </xsl:choose>
8037 </xsl:template>

```

`comp-space` This compensates for deficient space-stripping where significant white-space-only nodes in mixed content are removed by the parser. It tests to see if the immediately preceding sibling node (if any) is an element, and if so, outputs a space.

```

8038 <xsl:template name="compensate-space">
8039   <xsl:if test="preceding-sibling::node() and
8040               preceding-sibling::* and
8041               count(preceding-sibling::node()[1] |
8042                     preceding-sibling::*[1])=1">
8043     <xsl:text> </xsl:text>
8044   </xsl:if>
8045 </xsl:template>

```

`repeatarg` Generates $\$limit$ repeated TeX arguments when forming a generated command in a `<cmdsynopsis>`.

```

8046 <xsl:template name="repeatarg">
8047   <xsl:param name="limit"/>
8048   <xsl:param name="count">0</xsl:param>
8049   <xsl:text>0</xsl:text>
8050 </xsl:template>
8051 <xsl:template name="prefix"/>
8052 <xsl:template name="suffix"/>
8053 <xsl:if test="\$count<\$limit">
8054   <xsl:value-of select="\$prefix"/>
8055   <xsl:text>#</xsl:text>
8056   <xsl:value-of select="\$count + 1"/>
8057   <xsl:value-of select="\$suffix"/>
8058   <xsl:call-template name="repeatarg">
8059     <xsl:with-param name="limit" select="\$limit"/>
8060     <xsl:with-param name="count" select="\$count + 1"/>
8061   </xsl:call-template>
8062 </xsl:if>
8063 </xsl:template>

```

11.5.5 Copyright

`copyright` This template generates the copyright and boilerplate information block at the top of every extractable L^AT_EX file.

```

8064 <xsl:template name="copyright-statement">
8065   <xsl:param name="ftype"/>
8066   <xsl:text>% Transformed from </xsl:text>
8067   <xsl:value-of select="\$docname"/>
8068   <xsl:text>.xml by ClassPack db2dtx.xml
8069   % version </xsl:text>
8070   <xsl:value-of select="\$thisversion"/>
8071   <xsl:text> (</xsl:text>
8072   <xsl:value-of select="\$thisverdate"/>
8073   <xsl:text>) on </xsl:text>
8074   <xsl:value-of
8075     select="format-dateTime(current-dateTime(),
8076                             '[FNn] [D] [MNn] [Y] at [H]:[m]:[s]')"/>
8077   <xsl:text>&#xa;%&#xa;% </xsl:text>

```

```

8078     <xsl:value-of select="$docname"/>
8079     <xsl:text>.</xsl:text>
8080     <xsl:value-of select="$ftype"/>
8081     <xsl:text> is copyright © </xsl:text>
8082     <xsl:variable name="copyright-first">
8083       <xsl:for-each
8084         select="//db:info/db:revhistory/db:revision">
8085         <xsl:sort select="@version" order="descending"/>
8086         <xsl:if test="position()=last()">
8087           <xsl:value-of
8088             select="substring(db:date/@YYYY-MM-DD,1,4)"/>
8089         </xsl:if>
8090       </xsl:for-each>
8091     </xsl:variable>
8092     <xsl:value-of select="$copyright-first"/>
8093     <xsl:variable name="copyright-latest"
8094       select="substring($date,1,4)"/>
8095     <xsl:if test="$copyright-latest > $copyright-first">
8096       <xsl:text>-</xsl:text>
8097       <xsl:value-of select="$copyright-latest"/>
8098     </xsl:if>
8099     <xsl:text> by </xsl:text>
8100     <!-- RIGHTS -->
8101     <xsl:choose>
8102       <xsl:when test="//db:book/@audience='lppl'">
8103         <xsl:value-of
8104           select="normalize-space(//db:info//db:author[1]/
8105             db:personname/db:firstname)"/>
8106         <xsl:text> </xsl:text>
8107         <xsl:value-of
8108           select="normalize-space(//db:info//db:author[1]/
8109             db:personname/db:surname)"/>
8110         <xsl:text> &lt;</xsl:text>
8111         <xsl:value-of
8112           select="//db:info//db:author[1]/db:email"/>
8113         <xsl:text>>&#xa;</xsl:text>
8114         <xsl:text>%
8115 % This work may be distributed and/or modified under the
8116 % conditions of the LaTeX Project Public License, either
8117 % version 1.3 of this license or (at your option) any later
8118 % version. The latest version of this license is in:
8119 %
8120 %   http://www.latex-project.org/lppl.txt
8121 %
8122 % and version 1.3 or later is part of all distributions of
8123 % LaTeX version 2005/12/01 or later.
8124 %
8125 % This work has the LPPL maintenance status 'maintained'.
8126 % &#xa;</xsl:text>
8127       </xsl:when>
8128       <xsl:otherwise>
8129         <xsl:value-of
8130           select="normalize-space(
8131             //db:info/db:copyright/db:holder)"/>

```

```

8132         <xsl:text> &lt;/xsl:text>
8133         <xsl:value-of
8134             select="substring-after(
8135                 //db:info/db:copyright/db:holder
8136                 /@xlink:href,'mailto:')"/>
8137         <xsl:text>>&#xa;</xsl:text>
8138         <xsl:text>%
8139 % This work may not be copied or re-used without the express
8140 % written permission of the copyright holder[s]. Access to
8141 % this work is restricted to the copyright holder[s] and
8142 % their authorised employees, contractors, or agents.
8143 % &#xa;</xsl:text>
8144     </xsl:otherwise>
8145 </xsl:choose>
8146 <xsl:text>% The current maintainer of this work is </xsl:text>
8147 <xsl:choose>
8148     <xsl:when
8149         test="//db:info//db:author[@role='maintainer']">
8150         <xsl:for-each
8151             select="//db:info//db:author
8152                 [@role='maintainer']">
8153             <xsl:value-of
8154                 select="normalize-space(
8155                     db:personname/db:firstname)"/>
8156             <xsl:text> </xsl:text>
8157             <xsl:value-of
8158                 select="normalize-space(
8159                     db:personname/db:surname)"/>
8160             <xsl:text> &lt;/xsl:text>
8161             <xsl:value-of select="db:email"/>
8162         </xsl:for-each>
8163     </xsl:when>
8164     <xsl:otherwise>
8165         <xsl:value-of
8166             select="normalize-space(
8167                 //db:info//db:author/
8168                 db:personname/db:firstname)"/>
8169         <xsl:text> </xsl:text>
8170         <xsl:value-of
8171             select="normalize-space(
8172                 //db:info//db:author/
8173                 db:personname/db:surname)"/>
8174         <xsl:text> &lt;/xsl:text>
8175         <xsl:value-of
8176             select="//db:info//db:author/db:email"/>
8177     </xsl:otherwise>
8178 </xsl:choose>
8179 <xsl:text>>&#xa;</xsl:text>
8180 <xsl:text>%
8181 % This work consists of the files </xsl:text>
8182 <xsl:value-of select="$docname"/>
8183 <xsl:text>.dtx and </xsl:text>
8184 <xsl:value-of select="$docname"/>
8185 <xsl:text>.ins,</xsl:text>

```

```

8186     <xsl:if test="/db:book/db:part
8187                 [@xml:id='code']
8188                 /db:appendix">
8189     <xsl:text>&#xa;% the derived file</xsl:text>
8190     <xsl:if test="count(/db:book/db:part
8191                 [@xml:id='code']
8192                 /db:appendix)>0">
8193     <xsl:text>s</xsl:text>
8194     </xsl:if>
8195     <xsl:text> </xsl:text>
8196     <xsl:for-each
8197     select="db:book/db:part
8198           [@xml:id='code']
8199           /(db:chapter|db:appendix)[@xlink:href]">
8200     <xsl:value-of select="@xlink:href"/>
8201     <xsl:text>, </xsl:text>
8202     </xsl:for-each>
8203     </xsl:if>
8204     <xsl:text>&#xa;% and any other ancillary files </xsl:text>
8205     <xsl:text>listed in the MANIFEST.&#xa;</xsl:text>
8206     </xsl:template>

```

11.5.6 DocTeX text

`dtxttext` Output text for the `.dtx` file. Elide empty and white-space-only strings; remove leading TABs. An earlier version replaced an initial newline in `<programlisting>` content with a percent sign, but this has been lost in the workflow.

Strings containing newlines get output with a leading percent sign as DocTeX's normal armour. Underscores in filenames get replaced with their \LaTeX escape.

```

8207     <xsl:template name="dtxttext">
8208     <xsl:param name="text"/>
8209     <xsl:choose>
8210     <xsl:when test="normalize-space(.)='' ">
8211     <xsl:text></xsl:text>
8212     </xsl:when>
8213     <xsl:when test="starts-with($text,'&#x9;') ">
8214     <xsl:call-template name="dtxttext">
8215     <xsl:with-param name="text"
8216     select="substring($text,2)"/>
8217     </xsl:call-template>
8218     </xsl:when>
8219     <!-- if it starts with a newline AND
8220     it's in a <programlisting> element AND
8221     this is the first span of its type, prefix
8222     a percent -->
8223     <xsl:when test="starts-with($text,'&#xa;') and
8224     parent::db:programlisting and
8225     count(preceding-sibling::text())=0">
8226     <xsl:text>% </xsl:text>
8227     <xsl:call-template name="dtxttext">
8228     <xsl:with-param name="text"
8229     select="substring($text,2)"/>

```

```

8230     </xsl:call-template>
8231 </xsl:when>
8232 <!-- break strings containing newlines -->
8233 <xsl:when test="contains($text,'&#xa;')">
8234     <xsl:value-of
8235       select="substring-before($text,'&#xa;')"/>
8236     <xsl:choose>
8237       <xsl:when
8238         test="normalize-space(substring-after(
8239           $text,'&#xa;'))=''
8240           and
8241           parent::db:programlisting
8242           and
8243           count(following-sibling::text())=0">
8244         <xsl:text></xsl:text>
8245       </xsl:when>
8246       <xsl:otherwise>
8247         <xsl:text>&#xa;% </xsl:text>
8248         <xsl:call-template name="dtxtext">
8249           <xsl:with-param name="text"
8250             select="substring-after($text,'&#xa;')"/>
8251         </xsl:call-template>
8252       </xsl:otherwise>
8253     </xsl:choose>
8254 </xsl:when>
8255 <!-- convert underscores in filenames -->
8256 <xsl:when
8257   test="ancestor::db:filename[db:replaceable] and
8258     contains($text,'_')">
8259   <xsl:value-of select="substring-before($text,'_')"/>
8260   <xsl:text>\_</xsl:text>
8261   <xsl:call-template name="dtxtext">
8262     <xsl:with-param name="text"
8263       select="substring-after($text,'_')"/>
8264   </xsl:call-template>
8265 </xsl:when>
8266 <xsl:otherwise>
8267   <xsl:value-of select="$text"/>
8268 </xsl:otherwise>
8269 </xsl:choose>
8270 </xsl:template>

```

delogify Routine to replace T_EX/L_AT_EX logos with plaintext versions.

```

8271 <xsl:template name="delogify">
8272   <xsl:param name="string"/>
8273   <xsl:choose>
8274     <xsl:when test="contains($string,'\LaTeXe{')">
8275       <xsl:call-template name="delogify">
8276         <xsl:with-param name="string">
8277           <xsl:value-of
8278             select="substring-before($string,'\LaTeXe{')"/>
8279         <xsl:text>LaTeX2e</xsl:text>
8280       <xsl:value-of

```

```

8281         select="substring-after($string,'\LaTeX{}')"/>
8282     </xsl:with-param>
8283 </xsl:call-template>
8284 </xsl:when>
8285 <xsl:when test="contains($string,'\LaTeX{}')">
8286     <xsl:call-template name="delogify">
8287         <xsl:with-param name="string">
8288             <xsl:value-of
8289                 select="substring-before($string,'\LaTeX{}')"/>
8290             <xsl:text>LaTeX</xsl:text>
8291             <xsl:value-of
8292                 select="substring-after($string,'\LaTeX{}')"/>
8293             </xsl:with-param>
8294         </xsl:call-template>
8295     </xsl:when>
8296 <xsl:when test="contains($string,'\TeX{}')">
8297     <xsl:call-template name="delogify">
8298         <xsl:with-param name="string">
8299             <xsl:value-of
8300                 select="substring-before($string,'\TeX{}')"/>
8301             <xsl:text>TeX</xsl:text>
8302             <xsl:value-of
8303                 select="substring-after($string,'\TeX{}')"/>
8304             </xsl:with-param>
8305         </xsl:call-template>
8306     </xsl:when>
8307 <xsl:otherwise>
8308     <xsl:value-of select="$string"/>
8309 </xsl:otherwise>
8310 </xsl:choose>
8311 </xsl:template>

```

11.6 Modes

`insfile` Creates the `.ins` installer file, named after the main file. After the copyright block, it follows the standard pattern documented in Pakin (2015, section 2). The preamble is omitted when outputting a script, as it would have no meaning.

```

8312 <xsl:template match="db:info" mode="ins">
8313     <xsl:variable name="file">
8314         <xsl:value-of select="$docname"/>
8315         <xsl:text>.ins</xsl:text>
8316     </xsl:variable>
8317     <xsl:result-document format="textFormat" href="{ $file }">
8318         <xsl:call-template name="copyright-statement">
8319             <xsl:with-param name="ftype">
8320                 <xsl:text>ins</xsl:text>
8321             </xsl:with-param>
8322         </xsl:call-template>
8323         <xsl:text>%
8324         \input docstrip.tex
8325         \keepsilent
8326         \usedir{</xsl:text>

```

```

8327     <xsl:choose>
8328         <xsl:when test="//db:book/@xml:base">
8329             <xsl:value-of select="//db:book/@xml:base"/>
8330         </xsl:when>
8331         <xsl:otherwise>
8332             <xsl:text>tex/latex</xsl:text>
8333         </xsl:otherwise>
8334     </xsl:choose>
8335     <xsl:text>}&#xa;</xsl:text>
8336     <xsl:choose>
8337         <xsl:when test="$doctype='script'">
8338             <xsl:text>\nopreamble&#xa;</xsl:text>
8339         </xsl:when>
8340         <xsl:otherwise>
8341             <xsl:text>\preamble
8342
8343 This is a generated file.
8344
8345 Copyright © </xsl:text>
8346     <xsl:value-of
8347         select="format-date(db:copyright
8348             /db:year/@YYYY-MM-DD,'[Y]')"/>
8349     <!-- omit range if copyright is this year -->
8350     <xsl:if
8351         test="number(
8352             substring(
8353                 db:copyright/db:year/@YYYY-MM-DD,1,4))
8354         &lt; year-from-date(current-date())">
8355         <xsl:text>-</xsl:text>
8356         <xsl:value-of
8357             select="format-date(current-date(),'[Y]')"/>
8358     </xsl:if>
8359     <xsl:text> by </xsl:text>
8360     <xsl:value-of
8361         select="normalize-space(db:copyright
8362             /db:holder)"/>
8363     <xsl:text>&#xa;&#xa;</xsl:text>
8364     <xsl:for-each select="db:annotation/*">
8365         <xsl:call-template name="htrim">
8366             <xsl:with-param name="text">
8367                 <xsl:call-template name="lrtrim">
8368                     <xsl:with-param name="text" select="."/>
8369                 </xsl:call-template>
8370             </xsl:with-param>
8371         </xsl:call-template>
8372         <xsl:text>&#xa;&#xa;</xsl:text>
8373     </xsl:for-each>
8374     <xsl:text>\endpreamble&#xa;</xsl:text>
8375 </xsl:otherwise>
8376 </xsl:choose>

```

The following fragment is not active (commented out) as it appears to generate errors and is not needed.

```

8377      <!-- inactive due to errors
8378      <xsl:for-each
8379          select="//db:book/db:part[@xml:id='files']
8380              /db:chapter[db:programlisting]">
8381          <xsl:variable name="ext"
8382              select="substring-after(@xlink:href, '.')"/>
8383          <xsl:text>\declarepreamble\</xsl:text>
8384          <xsl:value-of select="@xml:id"/>
8385          <xsl:text>
8386 \DoubleperCent\space This is file </xsl:text>
8387          <xsl:value-of select="@xlink:href"/>
8388          <xsl:text>&#xa;\endpreamble&#xa;</xsl:text>
8389          <xsl:text>\declarepostamble\</xsl:text>
8390          <xsl:value-of select="@xml:id"/>
8391          <xsl:text>
8392 \DoubleperCent\space End of file </xsl:text>
8393          <xsl:value-of select="@xlink:href"/>
8394          <xsl:text>&#xa;\endpreamble&#xa;</xsl:text>
8395      </xsl:for-each>
8396      -->

```

Generate the extractor for the main file.

```

8397      <xsl:text>\generate{</xsl:text>
8398      <!-- 1. Generate the main file (cls or sty) first -->
8399      <xsl:text>\file{</xsl:text>
8400      <xsl:value-of select="$name"/>
8401      <xsl:text>.</xsl:text>
8402      <xsl:value-of select="$filetype"/>
8403      <xsl:text>}{\from{</xsl:text>
8404      <xsl:value-of select="$docname"/>
8405      <xsl:text>.dtx}{</xsl:text>
8406      <xsl:value-of select="$doctype"/>
8407      <xsl:text>}}&#xa;</xsl:text>

```

Add any file fragments from the documentation needing to be output on their own; these are identified as a <programlisting> with an ID, an @xlink:href, and @show set to "new".

```

8408      <!-- 2. Then any new files FROM WITHIN THE DOCUMENTATION (!)
8409           specially designated -->
8410      <xsl:for-each
8411          select="//db:programlisting
8412              [@xml:id
8413              and
8414              @xlink:show='new'
8415              and
8416              @xlink:href]">
8417          <xsl:variable name="ext"
8418              select="substring-after(@xlink:href, '.')"/>
8419          <xsl:choose>
8420              <xsl:when test="$ext='tex'">
8421                  <xsl:text>\usepreamble\</xsl:text>
8422                  <xsl:value-of select="@xml:id"/>

```

```

8423         <xsl:text>preamble\usepostamble\</xsl:text>
8424         <xsl:value-of select="@xml:id"/>
8425         <xsl:text>postamble</xsl:text>
8426     </xsl:when>
8427     <xsl:otherwise>
8428         <xsl:text>                </xsl:text>
8429         <xsl:text>\usepreamble\empty</xsl:text>
8430         <xsl:text>\usepostamble\empty</xsl:text>
8431     </xsl:otherwise>
8432 </xsl:choose>
8433 <xsl:text>&#xa;                \file{</xsl:text>
8434 <xsl:value-of select="@xlink:href"/>
8435 <xsl:text>}}{</xsl:text>
8436 <xsl:value-of select="$docname"/>
8437 <xsl:text>.dtx}}{</xsl:text>
8438 <xsl:value-of select="@xml:id"/>
8439 <xsl:text>}}&#xa;</xsl:text>
8440 </xsl:for-each>

```

Add any *annotated* ancillary files from code appendices.

```

8441 <!-- 3. Then the files with annotations -->
8442 <xsl:for-each
8443     select="/db:book/db:part[@xml:id='code']
8444         /db:appendix
8445         [descendant::db:programlisting
8446         and
8447         @xml:id
8448         and
8449         @xlink:href
8450         and
8451         not(@userlevel='cp')]">
8452 <!-- check omitted cp files are in manifest -->
8453 <xsl:if test="@userlevel='cp' and
8454             id('manifest')/db:simplelist/db:member
8455             [=current()/@xlink:href]">
8456     <xsl:message>
8457         <xsl:text>Warning: file </xsl:text>
8458         <xsl:value-of select="@xlink:href"/>
8459         <xsl:text>not being generated but in manifest</xsl:text>
8460     </xsl:message>
8461 </xsl:if>
8462 <xsl:variable name="ext"
8463     select="substring-after(@xlink:href, '.')"/>
8464 <xsl:choose>
8465     <xsl:when test="$ext='tex'">
8466         <xsl:text>                \usepreamble\</xsl:text>
8467         <xsl:value-of select="@xml:id"/>
8468         <xsl:text>preamble\usepostamble\</xsl:text>
8469         <xsl:value-of select="@xml:id"/>
8470         <xsl:text>postamble&#xa;</xsl:text>
8471     </xsl:when>
8472     <xsl:when test="$ext='cls' or $ext='sty'">
8473 <!--

```

```

8474     <xsl:text>                \usepreamble\usepostamble</xsl:text>
8475 -->
8476         </xsl:when>
8477         <xsl:otherwise>
8478             <xsl:text>                </xsl:text>
8479             <xsl:text>\usepreamble\empty</xsl:text>
8480             <xsl:text>\usepostamble\empty&#xa;</xsl:text>
8481         </xsl:otherwise>
8482     </xsl:choose>
8483     <xsl:text>                \file{</xsl:text>
8484     <xsl:value-of select="@xlink:href"/>
8485     <xsl:text>}}{\from{</xsl:text>
8486     <xsl:value-of select="$docname"/>
8487     <xsl:text>.dtx}}{</xsl:text>
8488     <xsl:value-of select="@xml:id"/>
8489     <xsl:text>}}&#xa;</xsl:text>
8490 </xsl:for-each>

```

Add any *unannotated* ancillary files from chapters in the files part.

```

8491     <!-- 4. Then the files extracted whole without annotation -->
8492     <xsl:for-each
8493         select="/db:book/db:part[@xml:id='files']
8494             /db:chapter
8495             [db:programlisting
8496             and
8497             @xml:id
8498             and @xlink:href]">
8499     <xsl:variable name="ext"
8500         select="substring-after(@xlink:href, '.')"/>
8501     <xsl:choose>
8502         <xsl:when test="$ext='TeX'">
8503             <xsl:text>                \usepreamble\</xsl:text>
8504             <xsl:value-of select="@xml:id"/>
8505             <xsl:text>preamble\usepostamble\</xsl:text>
8506             <xsl:value-of select="@xml:id"/>
8507             <xsl:text>postamble</xsl:text>
8508         </xsl:when>
8509         <xsl:otherwise>
8510             <xsl:text>                </xsl:text>
8511             <xsl:text>\usepreamble\empty</xsl:text>
8512             <xsl:text>\usepostamble\empty</xsl:text>
8513         </xsl:otherwise>
8514     </xsl:choose>
8515     <xsl:text>&#xa;                \file{</xsl:text>
8516     <xsl:value-of select="@xlink:href"/>
8517     <xsl:text>}}{\from{</xsl:text>
8518     <xsl:value-of select="$docname"/>
8519     <xsl:text>.dtx}}{</xsl:text>
8520     <xsl:value-of select="@xml:id"/>
8521     <xsl:text>}}&#xa;</xsl:text>
8522 </xsl:for-each>

```

Generate the console message to the user. A lot of time is taken here trying to get

it to self-adjust to the file name lengths so that it looks right.

```

8523         <xsl:text>}
8524 \obeyspaces
8525 \Msg{*****}
8526 \Msg{**                                     **}
8527 \Msg{** Read the documentation before using </xsl:text>
8528         <xsl:choose>
8529             <xsl:when
8530                 test="not(db:part[@xml:id='class']
8531                     /descendant::db:appendix[@xml:id])">
8532                 <xsl:text>this </xsl:text>
8533                 <xsl:value-of select="$doctype"/>
8534                 <xsl:text>.    </xsl:text>
8535             </xsl:when>
8536             <xsl:when
8537                 test="$doctype='cls' and
8538                     not(db:part[@xml:id='class']
8539                         /descendant::db:appendix[@xml:id]
8540                         [substring-after(@xlink:href, '.')!='cls'])">
8541                 <xsl:text>these classes. </xsl:text>
8542             </xsl:when>
8543             <xsl:when
8544                 test="$doctype='sty' and
8545                     not(db:part[@xml:id='class']
8546                         /descendant::db:appendix[@xml:id]
8547                         [substring-after(@xlink:href, '.')!='sty'])">
8548                 <xsl:text>these packages.</xsl:text>
8549             </xsl:when>
8550             <xsl:otherwise>
8551                 <xsl:text>these files.    </xsl:text>
8552             </xsl:otherwise>
8553         </xsl:choose>
8554         <xsl:text>**}
8555 \Msg{**                                     **}
8556 \Msg{** To finish the installation you have to move the    **}
8557 \Msg{** following file</xsl:text>
8558         <xsl:if
8559             test="count(/db:book/db:part[@xml:id='files']
8560                 /db:chapter[db:programlisting and
8561                     @xml:id and @xlink:href]) +
8562                 count(/db:book/db:part[@xml:id='code']
8563                     /db:appendix[db:programlisting and
8564                         @xml:id and @xlink:href]) + 1
8565                 &gt; 1">
8566             <xsl:text>s</xsl:text>
8567         </xsl:if>
8568         <xsl:text> into a directory searched by TeX: </xsl:text>
8569         <xsl:if
8570             test="count(/db:book/db:part[@xml:id='files']
8571                 /db:chapter[db:programlisting and
8572                     @xml:id and @xlink:href]) +
8573                 count(/db:book/db:part[@xml:id='code']
8574                     /db:appendix[db:programlisting and

```

```

8575                                     @xml:id and @xlink:href]) + 1
8576                                     = 1">
8577         <xsl:text> </xsl:text>
8578     </xsl:if>
8579     <xsl:text>**}
8580 \Msg{**                                     **}
8581 </xsl:text>
8582     <!-- class or package first -->
8583     <xsl:text>\Msg{**     </xsl:text>
8584     <xsl:value-of
8585         select="/db:book/db:part
8586             [@xml:id='code']
8587             /db:chapter[1]/@xlink:href"/>
8588     <xsl:variable name="spacetaken"
8589         select="4 +
8590             string-length(/db:book/db:part[@xml:id='code']
8591                 /db:chapter[1]/@xlink:href)
8592             + 4"/>
8593     <xsl:value-of
8594         select="for $s in (1 to
8595             xs:integer((50 - $spacetaken) div 2))
8596             return '&#x20;'" />
8597     <xsl:text>     **}</xsl:text>
8598     <!-- remaining files -->
8599     <xsl:for-each
8600         select="/db:book/db:part[@xml:id='files']
8601             /db:chapter[descendant::db:programlisting
8602                 and
8603                 @xml:id and @xlink:href]
8604             |
8605             /db:book/db:part[@xml:id='code']
8606             /db:appendix[descendant::db:programlisting
8607                 and
8608                 @xml:id and @xlink:href]">
8609     <xsl:text>&#xa;\Msg{**     </xsl:text>
8610     <xsl:value-of select="@xlink:href"/>
8611     <xsl:variable name="spacetaken"
8612         select="string-length(@xlink:href)"/>
8613     <xsl:value-of
8614         select="for $s in (1 to
8615             xs:integer((46 - $spacetaken) div 2))
8616             return '&#x20;'" />
8617     <xsl:if test="($spacetaken div 2) !=
8618         (xs:integer($spacetaken) div 2)">
8619         <xsl:text>     </xsl:text>
8620     </xsl:if>
8621     <xsl:text>     **}</xsl:text>
8622 </xsl:for-each>
8623 <xsl:text>
8624 \Msg{**                                     **}
8625 \Msg{** To produce the documentation run the file     **}
8626 \Msg{** </xsl:text>
8627     <xsl:value-of select="$docname"/>
8628     <xsl:text>.dtx through </xsl:text>

```

```

8629      <!-- space available is 50 chars less the word 'through' (7)
8630           and the two spaces either side, less the width of the
8631           class/package name plus 4 for the extension, less the
8632           width of the program names and their spaces -->
8633      <xsl:choose>
8634        <xsl:when test="//db:part[@conformance='lualatex']">
8635          <xsl:text>XeLaTeX and biber</xsl:text>
8636          <xsl:variable name="spacetaken"
8637            select="string-length($docname) + 4 + 9 + 17"/>
8638          <xsl:value-of
8639            select="for $s in (1 to
8640              xs:integer((50 - $spacetaken) div 2))
8641              return '&#x20;'" />
8642        </xsl:when>
8643        <xsl:when test="//db:part[@conformance='xelatex']">
8644          <xsl:text>XeLaTeX and biber</xsl:text>
8645          <xsl:variable name="spacetaken"
8646            select="string-length($docname) + 4 + 9 + 17"/>
8647          <xsl:value-of
8648            select="for $s in (1 to
8649              xs:integer((50 - $spacetaken) div 2))
8650              return '&#x20;'" />
8651        </xsl:when>
8652        <xsl:otherwise>
8653          <xsl:text>pdfLaTeX and BiBTeX</xsl:text>
8654          <xsl:variable name="spacetaken"
8655            select="string-length($docname) + 4 + 9 + 19"/>
8656          <xsl:value-of
8657            select="for $s in (1 to
8658              xs:integer((50 - $spacetaken) div 2))
8659              return ' '" />
8660        </xsl:otherwise>
8661      </xsl:choose>
8662      <xsl:text>  **}
8663      \Msg{**                                **}
8664      \Msg{** Happy LaTeXing!                **}
8665      \Msg{**                                **}
8666      \Msg{*****}
8667      \endbatchfile
8668    </xsl:text>
8669  </xsl:result-document>
8670 </xsl:template>

```

insfile **TODO: what about stuff in the data part?**

files When extracting unannotated files (eg examples or configs), elide the titles in parts and chapters. The sole other component of such chapters, apart from the title, must be the `<programlisting>` element holding the code. This is output as a DocTeX macrocode block.

```

8671    <!--
8672    <xsl:template

```

```

8673     match="db:part/db:title |
8674         db:chapter/db:title |
8675         db:chapter/db:subtitle |
8676         db:chapter/db:para" mode="files"/>
8677 -->
8678
8679 <xsl:template match="db:part/db:title" mode="files">
8680   <xsl:text>% \section{</xsl:text>
8681   <xsl:apply-templates mode="files"/>
8682   <xsl:text>}\label{</xsl:text>
8683   <xsl:value-of select="../@xml:id"/>
8684   <xsl:text>}&#xa;</xsl:text>
8685 </xsl:template>
8686
8687 <xsl:template match="db:chapter" mode="files">
8688   <xsl:apply-templates mode="files"/>
8689 </xsl:template>
8690
8691 <xsl:template match="db:chapter/db:title" mode="files">
8692   <xsl:text>% \subsection{</xsl:text>
8693   <xsl:apply-templates/>
8694   <xsl:text>}\label{</xsl:text>
8695   <xsl:value-of select="../@xml:id"/>
8696   <xsl:text>}&#xa;</xsl:text>
8697 </xsl:template>
8698
8699 <xsl:template match="db:warning" mode="files">
8700   <xsl:call-template name="sidewarn"/>
8701 </xsl:template>
8702
8703 <xsl:template match="db:chapter/db:para
8704     |
8705     db:part/db:para" mode="files">
8706   <xsl:text>% </xsl:text>
8707   <xsl:apply-templates/>
8708   <xsl:text>&#xa;</xsl:text>
8709 </xsl:template>
8710
8711 <!--
8712 emphasis
8713 filename
8714 package
8715 productname
8716 tag
8717 xref
8718 -->
8719
8720 <xsl:template match="db:programlisting" mode="files">
8721   <xsl:choose>
8722     <xsl:when test="ancestor::db:part[@xml:id='files']">
8723       <xsl:text>% \iffalse&#xa;</xsl:text>
8724     </xsl:when>
8725     <xsl:otherwise>
8726       <xsl:text>%   </xsl:text>

```

```

8727     <xsl:text>\begin</xsl:text>
8728     <xsl:text>{macrocode}&#xa;</xsl:text>
8729     </xsl:otherwise>
8730 </xsl:choose>
8731 <!-- create the latexdoc tag -->
8732 <xsl:text>%&lt;*</xsl:text>
8733 <xsl:value-of select="parent::db:chapter/@xml:id"/>
8734 <xsl:text>>&#xa;</xsl:text>
8735 <xsl:call-template name="lrtrim">
8736   <xsl:with-param name="text"
8737     select="unparsed-text(concat($docname,'/',@xlink:href))"/>
8738 </xsl:call-template>
8739 <!-- end tag -->
8740 <xsl:text>&#xa;%&lt;/</xsl:text>
8741 <xsl:value-of select="parent::db:chapter/@xml:id"/>
8742 <xsl:text>>&#xa;</xsl:text>
8743 <xsl:choose>
8744   <xsl:when test="ancestor::db:part[@xml:id='files']">
8745     <xsl:text>% \fi&#xa;</xsl:text>
8746   </xsl:when>
8747   <xsl:otherwise>
8748     <xsl:text>%    </xsl:text>
8749     <xsl:text>\end</xsl:text>
8750     <xsl:text>{macrocode}&#xa;</xsl:text>
8751   </xsl:otherwise>
8752 </xsl:choose>
8753 </xsl:template>

```

11.7 Creating the web page

This named template creates the web page (index.html) for the class or package to go on the author's web site.

index Currently this reflects the style used on the Silmaril site.

```

8754 <xsl:variable name="TeX">
8755   <span class="TeX">T<span
8756     class="E">E</span>X</span>
8757 </xsl:variable>
8758 <xsl:variable name="LaTeX">
8759   <span class="LaTeX">L<span
8760     class="A">A</span><span
8761     class="TeX">T<span
8762     class="E">E</span>X</span></span>
8763 </xsl:variable>
8764 <xsl:template name="webindex">
8765   <xsl:message>
8766     <xsl:text>Making index.html for web site</xsl:text>
8767   </xsl:message>
8768   <xsl:result-document format="htmlFormat" href="index.html">
8769     <html>
8770       <head>
8771         <title>LaTeX Packages and Document Classes

```

```

8772         from Silmaril</title>
8773         <meta charset="utf-8"/>
8774         <meta name="viewport"
8775             content="width=device-width,initial-scale=1,
8776                 user-scalable=no"/>
8777         <link rel="stylesheet" href="../assets/css/main.css"/>
8778         <noscript>
8779             <link rel="stylesheet"
8780                 href="../assets/css/noscript.css"/>
8781         </noscript>
8782     </head>
8783     <body class="is-preload">
8784         <section id="top" class="wrapper style2">
8785             <header class="major">
8786                 <h2>
8787                     <xsl:copy-of select="$LaTeX"/>
8788                     <xsl:text> style packages and document classes
8789                     from Silmaril</xsl:text>
8790                 </h2>
8791                 <ul class="icons labeled">
8792                     <li>
8793                         <span class="icon solid fa-book">
8794                             <span class="label">
8795                                 <a href="#downloads">Index</a>
8796                             </span>
8797                         </span>
8798                     </li>
8799                     <li>
8800                         <span class="icon solid fa-book-open">
8801                             <span class="label">
8802                                 <a href="#more">About</a>
8803                             </span>
8804                         </span>
8805                     </li>
8806                 </ul>
8807             </header>
8808             <div class="inner alt">
8809                 <section class="spotlight" id="{ $docname }">
8810                     <div class="image">
8811                         
8812                     </div>
8813                     <div class="content">
8814                         <h3>
8815                             <xsl:text>The </xsl:text>
8816                             <a href="http://ctan.org/pkg/{ $docname }">
8817                                 <xsl:value-of select="{ $docname }"/>
8818                             </a>
8819                             <xsl:text> package</xsl:text>
8820                         </h3>
8821                         <h4 class="subtitle">
8822                             <xsl:text>v </xsl:text>
8823                             <xsl:value-of select="/db:book/@version"/>
8824                             <xsl:text>.</xsl:text>
8825                             <xsl:value-of select="/db:book/@revision"/>

```

```

8826         <xsl:text> of </xsl:text>
8827         <xsl:value-of
8828           select="/db:book/db:info
8829                 /db:revhistory/db:revision
8830                 [@version=max(parent::db:revhistory
8831                 /db:revision/@version)]
8832                 /db:date/@YYYY-MM-DD"/>
8833         <xsl:text> </xsl:text>
8834         <span class="reversed">⊗</span>
8835         <xsl:text> </xsl:text>
8836         <xsl:value-of
8837           select="format-date($firstverdate,'[Y]')"/>
8838         <xsl:text> </xsl:text>
8839         <xsl:value-of
8840           select="format-date(current-date(),'[Y]')"/>
8841         <xsl:text> </xsl:text>
8842         <a href="www.latex-project.org/lppl/lppl-1-3c/">LPPL
8843           1.3c</a>
8844       </h4>
8845       <p>
8846         <xsl:value-of select="/db:book/db:info/db:title"/>
8847       </p>
8848       <div class="table-wrapper">
8849         <table>
8850           <thead>
8851             <tr>
8852               <th>File</th>
8853               <th>Description</th>
8854             </tr>
8855           </thead>
8856           <tbody>
8857             <!-- ZIP -->
8858             <tr>
8859               <td>
8860                 <tt>
8861                   <a href="{ $docname }-{ /db:book/@version }. {
8862                     /db:book/@revision }.zip">
8863                     <xsl:value-of
8864                       select="$docname"/>
8865                     <xsl:text>-</xsl:text>
8866                     <xsl:value-of
8867                       select="/db:book/@version"/>
8868                     <xsl:text>.</xsl:text>
8869                     <xsl:value-of
8870                       select="/db:book/@revision"/>
8871                     <xsl:text>.zip</xsl:text>
8872                   </a>
8873                 </tt>
8874               </td>
8875               <td>Zip file of the CTAN distribution</td>
8876             </tr>
8877             <!-- TDS ZIP -->
8878             <tr>
8879               <td>

```

```

8880         <tt>
8881             <a href="{ $docname }-{ /db:book/@version }. {
8882                 /db:book/@revision }.tds.zip">
8883                 <xsl:value-of
8884                     select="$docname"/>
8885                 <xsl:text>-</xsl:text>
8886                 <xsl:value-of
8887                     select="/db:book/@version"/>
8888                 <xsl:text>.</xsl:text>
8889                 <xsl:value-of
8890                     select="/db:book/@revision"/>
8891                 <xsl:text>.tds.zip</xsl:text>
8892             </a>
8893         </tt>
8894     </td>
8895     <td>TeX Directory Structure (TDS) zipfile</td>
8896 </tr>
8897 <!-- DTX -->
8898 <tr>
8899     <td>
8900         <tt>
8901             <a href="{ $docname }.dtx">
8902                 <xsl:value-of select="$docname"/>
8903                 <xsl:text>.dtx</xsl:text>
8904             </a>
8905         </tt>
8906     </td>
8907     <td>
8908         <xsl:text>Doc</xsl:text>
8909         <xsl:copy-of select="$TeX"/>
8910         <xsl:text> combined code and
8911         documentation</xsl:text>
8912     </td>
8913 </tr>
8914 <!-- INS -->
8915 <tr>
8916     <td>
8917         <tt>
8918             <a href="{ $docname }.ins">
8919                 <xsl:value-of select="$docname"/>
8920                 <xsl:text>.ins</xsl:text>
8921             </a>
8922         </tt>
8923     </td>
8924     <td>
8925         <xsl:text>Doc</xsl:text>
8926         <xsl:copy-of select="$TeX"/>
8927         <xsl:text> installer</xsl:text>
8928     </td>
8929 </tr>
8930 <!-- PDF -->
8931 <tr>
8932     <td>
8933         <tt>

```

```

8934         <a href="{ $docname }.pdf">
8935             <xsl:value-of select="$docname"/>
8936             <xsl:text>.pdf</xsl:text>
8937         </a>
8938     </tt>
8939 </td>
8940 <td>
8941     <xsl:text>PDF documentation</xsl:text>
8942 </td>
8943 </tr>
8944 <!-- CLS/STY -->
8945 <tr>
8946     <td>
8947         <tt>
8948             <a
8949                 href="{ $docname }.{/db:book/@userlevel}">
8950                 <xsl:value-of select="$docname"/>
8951                 <xsl:text>.</xsl:text>
8952                 <xsl:value-of
8953                     select="/db:book/@userlevel"/>
8954             </a>
8955         </tt>
8956     </td>
8957     <td>
8958         <xsl:text>the </xsl:text>
8959         <xsl:value-of select="/db:book/@arch"/>
8960         <xsl:text> file</xsl:text>
8961     </td>
8962 </tr>
8963 <!-- PNG -->
8964 <!-- convert \
8965     -density 300 <docname>.pdf[0] \
8966     -quality 100 \
8967     -flatten \
8968     -sharpen 0x1.0 <docname>.png
8969 -->
8970 <tr>
8971     <td>
8972         <tt>
8973             <a href="{ $docname }.png">
8974                 <xsl:value-of select="$docname"/>
8975                 <xsl:text>.png</xsl:text>
8976             </a>
8977         </tt>
8978     </td>
8979     <td>associated image</td>
8980 </tr>
8981 <!-- extractables -->
8982 <xsl:if
8983     test="/db:book/db:part[@xml:id='files']">
8984     <tr>
8985         <xsl:for-each
8986             select="/db:book/db:part
8987                 [@xml:id='files']

```

```

8988         /db:chapter[@xml:id
8989         and
8990         @xlink:href
8991         and
8992         db:programlisting]">
8993     <td>
8994         <tt>
8995             <a href="{@xlink:href}">
8996                 <xsl:value-of
8997                     select="@xlink:href"/>
8998             </a>
8999         </tt>
9000     </td>
9001     <td>
9002         <xsl:value-of select="@role"/>
9003     </td>
9004 </xsl:for-each>
9005 </tr>
9006 </xsl:if>
9007 <xsl:if
9008     test="/db:book/db:info/db:cover
9009         /db:constraintdef[@xml:id='manifest']">
9010 <xsl:for-each
9011     select="/db:book/db:info/db:cover/
9012         db:constraintdef
9013         [@xml:id='manifest']/
9014         db:simplelist/db:member">
9015 <tr>
9016     <td>
9017         <tt>
9018             <a href="{.}">
9019                 <xsl:value-of select="."/>
9020             </a>
9021         </tt>
9022     </td>
9023     <td>
9024         <xsl:value-of select="@role"/>
9025     </td>
9026 </tr>
9027 </xsl:for-each>
9028 </xsl:if>
9029 </tbody>
9030 </table>
9031 </div>
9032 <p>
9033     <xsl:text>Use the CTAN zip file with
9034     automated installers like </xsl:text>
9035     <em>tlmgr</em>
9036     <xsl:text> and MiK</xsl:text>
9037     <xsl:copy-of select="$TeX"/>
9038     <xsl:text>. Use the TDS zip file when
9039     installing to your </xsl:text>
9040     <a href="{concat('http://latex.silmaril.ie',
9041         '/formattinginformation/personal.html')}">

```

```

9042         <xsl:text>Personal </xsl:text>
9043         <xsl:copy-of select="$TeX"/>
9044         <xsl:text> Directory</xsl:text>
9045     </a>
9046     <xsl:text>. The individual DTX and INS files are
9047     for manual installation.</xsl:text>
9048 </p>
9049 <ul class="icons">
9050     <li>
9051         <a href="MANIFEST">MANIFEST</a>
9052     </li>
9053     <li>
9054         <a href="VERSION">VERSION</a>
9055     </li>
9056     <li>
9057         <a href="README.md">README.md</a>
9058     </li>
9059     <li class="generated">
9060         <xsl:text>ClassPack v </xsl:text>
9061         <xsl:value-of select="$thisversion"/>
9062         <br/>
9063         <xsl:value-of select="format-date(current-date(),
9064             '[D01]-[M01]-[Y03]')"/>
9065     </li>
9066 </ul>
9067 <ul class="actions">
9068     <li>
9069         <a class="button small primary scrolly"
9070         href="#downloads">Downloads</a>
9071     </li>
9072     <li>
9073         <a class="button small scrolly"
9074         href="#more">Learn More</a>
9075     </li>
9076     <li>
9077         <a class="button small scrolly"
9078         href="#contact">Contact us</a>
9079     </li>
9080     <li>
9081         <a class="button small scrolly"
9082         href="#header">Home</a>
9083     </li>
9084 </ul>
9085 </div>
9086 </section>
9087 </div>
9088 </section>
9089 <footer id="footer">
9090     <ul class="icons">
9091     <li>
9092         <a href="news:comp.text.tex"
9093         title="Usenet support"
9094         style="font-family:monospace;"
9095         class="icon solid fa-newspaper">

```

```

9096         <span class="label">comp.text.tex</span>
9097     </a>
9098 </li>
9099 <li>
9100     <a href="https://mastodon.ie/@latex"
9101         title="Talk to @latex on Mastodon"
9102         class="icon brands fa-deskpro">
9103         <span class="label">Mastodon</span>
9104     </a>
9105 </li>
9106 <li>
9107     <a href="https://twitter.com/silmaril_ie"
9108         title="Talk to @silmaril_ie on Twitter"
9109         class="icon brands fa-twitter">
9110         <span class="label">Twitter</span>
9111     </a>
9112 </li>
9113 <li>
9114     <a
9115         href="mailto:latex@silmaril.ie?subject=LaTeX support"
9116         style="font-family:monospace;"
9117         title="Email latex@silmaril.ie"
9118         class="icon solid fa-envelope-open">
9119         <span class="label">latex@silmaril.ie</span>
9120     </a>
9121 </li>
9122 </ul>
9123 <p class="copyright">
9124     <xsl:text>© </xsl:text>
9125     <xsl:value-of select="format-date($firstverdate,'[Y]')"/>
9126     <xsl:text> </xsl:text>
9127     <xsl:value-of select="format-date(current-date(),'[Y]')"/>
9128     <xsl:text> </xsl:text>
9129     <a href="http://www.silmaril.ie/">Silmaril Consultants</a>
9130     <xsl:text>. Design: </xsl:text>
9131     <a href="http://html5up.net">HTML5 UP</a>
9132 </p>
9133 </footer>
9134 <script src="../../assets/js/jquery.min.js"/>
9135 <script src="../../assets/js/jquery.scrolly.min.js"/>
9136 <script src="../../assets/js/browser.min.js"/>
9137 <script src="../../assets/js/breakpoints.min.js"/>
9138 <script src="../../assets/js/util.js"/>
9139 <script src="../../assets/js/main.js"/>
9140 </body>
9141 </html>
9142 </xsl:result-document>
9143 </xsl:template>

```

11.8 Ending

end End of the XSLT script.

```
9144 <xsl:template match="db:keywordset | db:keyword"/>
9145 </xsl:stylesheet>
```

The command used to manage the splitting of the file into cpdoc fragments is

```
chunk -p cpdoc db2dtx.xml; mv -f db2dtx*frag classpack
```

12 The db2bibtex.xsl script

The templates for converting reference data to B_IB_TE_X are kept in a separate file, `db2bibtex.xsl`, because they can be reused with any XSLT script handling *DocBook* documents which use a bibliography. This file is `<xsl:include>d` in [section 14.1 on page 355](#).

`db2bibtexscript-comment` Identify the file origin, date, time, etc.

```
1 <!-- The db2bibtex.xsl script (from the classpack package v.1.28, 2024-02-21) -->
2 <!-- Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16 -->
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

12.1 XML Declaration, namespaces, and setup

The namespaces are the same as for the calling routine, `db2dtx.xsl`.

`init` Set up the namespaces, the same as for the main script.

```
3 <xsl:stylesheet
4   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
5   xmlns:db="http://docbook.org/ns/docbook"
6   xmlns:xlink="http://www.w3.org/1999/xlink"
7   xmlns:xs="http://www.w3.org/2001/XMLSchema"
8   version="3.0">
```

`vars` We will need to test for apostrophes preceding abbreviated years in conference titles, so for simplicity we define it as a variable so it can be used in an argument delimited by apostrophes in an attribute delimited by double quotes, containing an `'`.

```
9   <xsl:variable name="apostrophe">'</xsl:variable>
```

12.1.1 Citations

`db:biblioref` The `<biblioref>` element type is used for all citations, and passed directly to the `makeref` named template. Attributes are [ab]used to provide for the features available in B_IB_TE_X and *biblatex*.

```
10 <xsl:template match="db:biblioref">
11   <xsl:call-template name="compensate-space"/>
12   <xsl:call-template name="makeref"/>
13 </xsl:template>
```

`makeref` Citations are formed by the named template `makeref`. It expects values for attributes on the calling `<biblioref>` element as detailed in ‘`<biblioref>`’, the second item in the list in [section 6.4 on page 65](#).

`makeref` This named template deals with the exceptions in handling citations, and the variations allowed. There are no checks for matching `<IDREF>` and `<ID>` values, as these have already been tested by the XML parser.

```
14 <xsl:template name="makeref">
15 <xsl:choose>
```

Direct commands (`multicite` and `fullcite`)

`multicite` The first test is for citing multiple sources: in XML, multiple `<IDREFS>` tokens are delimited by spaces, so these need converting to the commas required by \LaTeX . Multiple references at a single point always use the `\parencite` command, which encloses the entire thing in a single set of parentheses.

```
16 <xsl:when
17 test="contains(normalize-space(@linkend), ' ')">
18 <xsl:text>\parencite{</xsl:text>
19 <xsl:value-of
20 select="translate(
21 normalize-space(@linkend), ' ', ',')"/>
22 <xsl:text>}</xsl:text>
23 </xsl:when>
```

`fullcite` Full citation (long-form reference) is likewise handled by a command, `\fullcite`. The use of partial references in footnoted citations is not supported in *ClassPack*.

```
24 <xsl:when test="@xrefstyle='full'">
25 <xsl:text>\fullcite{</xsl:text>
26 <xsl:value-of select="@linkend"/>
27 <xsl:text>}</xsl:text>
28 </xsl:when>
```

Computed citations (full title, short title, and authors)

`citedetail` All remaining forms are handled by testing the actual reference data in the relevant `<biblioentry>` element.

```
29 <xsl:otherwise>
30 <xsl:variable name="target"
31 select="//db:biblioentry
32 [@xml:id=current()/@linkend]"/>
33 <xsl:choose>
```

`citetitle` Citations of titles handled here are *full* titles: the default `\citetitle` in the next section provides only the short title.

Note that this is *different* from the `\citetitle` element type of *DocBook* in [section 11.4.5 on page 173](#) which cites the `<title>` element of the entry but also allows a manually-typed title (ie one for which there is no bibliographic entry in the document).

```
34 <xsl:when test="@xrefstyle='title'
35 or
```

```

36             @xrefstyle='titlecite'"">
37     <xsl:choose>
38       <xsl:when
39         test="$target/@xreflabel='article' or
40             $target/@xreflabel='inbook' or
41             $target/@xreflabel='incollection' or
42             $target/@xreflabel='inreference'"">
43         <xsl:text>`</xsl:text>
44         <xsl:apply-templates
45           select="($target/descendant::db:title)
46             [1]/node()"/>
47         <xsl:text>'</xsl:text>
48       </xsl:when>
49       <xsl:otherwise>
50         <xsl:text>\emph{</xsl:text>
51         <xsl:apply-templates
52           select="($target/descendant::db:title)
53             [1]/node()"/>
54         <xsl:text>}</xsl:text>
55       </xsl:otherwise>
56     </xsl:choose>
57     <xsl:if test="@xrefstyle='titlecite'"">
58       <xsl:text>~(\citeyear{</xsl:text>
59       <xsl:value-of select="@linkend"/>
60       <xsl:text>}</xsl:text>
61     </xsl:if>
62   </xsl:when>

```

`citeshort` Separate citation of the short title is done by forming the citation manually.

```

63     <xsl:when
64       test="@xrefstyle='shorttitle'"">
65     <xsl:choose>
66       <xsl:when
67         test="$target/@xreflabel='article' or
68             $target/@xreflabel='inbook' or
69             $target/@xreflabel='incollection' or
70             $target/@xreflabel='inreference'"">
71         <xsl:text>`</xsl:text>
72         <xsl:choose>
73           <xsl:when
74             test="$target
75               /descendant::db:titleabbrev">
76             <xsl:apply-templates
77               select="($target
78                 /descendant::db:titleabbrev)
79                 [1]/node()"/>
80           </xsl:when>
81           <xsl:otherwise>
82             <xsl:apply-templates
83               select="($target
84                 /descendant::db:title)
85                 [1]/node()"/>
86           </xsl:otherwise>

```

```

87         </xsl:choose>
88         <xsl:text>'</xsl:text>
89     </xsl:when>
90     <xsl:otherwise>
91         <xsl:text>\emph{</xsl:text>
92         <xsl:choose>
93             <xsl:when
94                 test="$target
95                     /descendant::db:titleabbrev">
96                 <xsl:apply-templates
97                     select="($target
98                         /descendant::db:titleabbrev)
99                     [1]/node()"/>
100             </xsl:when>
101             <xsl:otherwise>
102                 <xsl:apply-templates
103                     select="($target
104                         /descendant::db:title)
105                     [1]/node()"/>
106             </xsl:otherwise>
107         </xsl:choose>
108         <xsl:text>}</xsl:text>
109     </xsl:otherwise>
110 </xsl:choose>
111 <xsl:if test="@xrefstyle='titlecite'">
112     <xsl:text>~(\citeyear{</xsl:text>
113     <xsl:value-of select="@linkend"/>
114     <xsl:text>})</xsl:text>
115 </xsl:if>
116 </xsl:when>

```

`citeauthors` **Author citation** just lists the authors in order.

```

117     <xsl:when
118         test="@xrefstyle='author' or
119             @xrefstyle='authors'">
120     <xsl:for-each
121         select="$target//db:author">
122     <xsl:if
123         test="preceding-sibling::db:author">
124     <xsl:choose>
125         <xsl:when
126             test="following-sibling::db:author">
127         <xsl:if
128             test="count(..db:author)>2">
129             <xsl:text>,</xsl:text>
130         </xsl:if>
131         </xsl:when>
132     <xsl:otherwise>
133         <xsl:if test="count(..db:author)>2">
134             <xsl:text>,</xsl:text>
135         </xsl:if>
136         <xsl:text> and </xsl:text>
137     </xsl:otherwise>

```

```

138         </xsl:choose>
139     </xsl:if>
140     <xsl:apply-templates
141         select="db:personname
142             /db:firstname/node() |
143             db:orgname/node() |
144             db:personname
145             /db:othername/node()" />
146     <xsl:text> </xsl:text>
147     <xsl:value-of
148         select="db:personname
149             /db:surname/node() |
150             db:orgdiv/node()" />
151 </xsl:for-each>
152 </xsl:when>

```

citesurnames Surname-only citation likewise.

citesurnames **TODO: Add punctuation and spacing for multiple authors**

```

153     <xsl:when
154         test="@xrefstyle='shortauthor' or
155             @xrefstyle='shortauthors'">
156     <xsl:for-each
157         select="$target//db:author
158             /db:personname">
159         <xsl:value-of
160             select="db:surname" />
161     </xsl:for-each>
162 </xsl:when>

```

Standard commands (cite, parencite, textcite)

citeother Everything else is handled by the standard commands (\cite, \parencite, and \textcite).

```

163     <xsl:otherwise>
164     <xsl:if
165         test="(//db:bibliography[@arch='biblatex']
166             or
167             //db:bibliolist[@arch='biblatex'])
168             and @xrefstyle='field'
169             and contains(@remap,'title')">
170         <xsl:text>\emph{</xsl:text>
171     </xsl:if>
172     <xsl:text>\</xsl:text>
173     <xsl:choose>
174     <xsl:when
175         test="(//db:bibliography[@arch='biblatex']
176             or
177             //db:bibliolist[@arch='biblatex'])
178             and
179             (@xrefstyle='text'

```

```

180             or
181             @xrefstyle='paren'
182             or
183             @xrefstyle='author(year)'
184             or
185             @xrefstyle='foot')">
186         <xsl:value-of select="@xrefstyle"/>
187     </xsl:when>
188     <xsl:when
189         test="//db:bibliography[@arch='biblatex']
190             or
191             //db:bibliolist[@arch='biblatex'])
192         and not(@xrefstyle)">
193         <xsl:text>paren</xsl:text>
194     </xsl:when>
195     <xsl:when
196         test="ancestor::db:section and @xrefstyle">
197         <xsl:value-of select="@xrefstyle"/>
198     </xsl:when>
199 </xsl:choose>
200 <xsl:text>cite</xsl:text>
201 <xsl:choose>
202     <xsl:when
203         test="//db:bibliography[@arch='biblatex']
204             or
205             //db:bibliolist[@arch='biblatex'])
206         and @xrefstyle='field' and @remap">
207         <xsl:text>field</xsl:text>
208     </xsl:when>
209     <xsl:when
210         test="//db:bibliography[@arch='biblatex']
211             or
212             //db:bibliolist[@arch='biblatex'])
213         and @xrefstyle='citetitle'">
214         <xsl:text>title</xsl:text>
215     </xsl:when>
216     <xsl:when
217         test="//db:bibliography
218             [@arch='bibtex' or not(@arch)]
219             and
220             @xrefstyle='text'">
221         <xsl:text>A</xsl:text>
222     </xsl:when>
223 </xsl:choose>

```

precite If @role is given explicitly, make it the pre-citation value.

```

224     <xsl:if test="@role">
225         <xsl:text>[</xsl:text>
226     <xsl:choose>
227         <xsl:when test="@role='firstname'">
228             <xsl:value-of
229                 select="//db:bibliography
230                     /db:biblioentry

```

```

231             [@xml:id=current()/@linkend]
232             //db:author[1]//db:firstname"/>
233         </xsl:when>
234         <xsl:otherwise>
235             <xsl:value-of select="@role"/>
236         </xsl:otherwise>
237     </xsl:choose>
238     <xsl:text>]</xsl:text>
239 </xsl:if>

```

`citeunits` Units (pages, chapters, etc) must be done as the optional argument.

```

240     <xsl:if test="(@units and @begin)
241                 or @annotations">
242         <xsl:if test="not(@role)">
243             <xsl:text>[]</xsl:text>
244         </xsl:if>
245         <xsl:text>[</xsl:text>
246         <xsl:value-of select="@units"/>
247         <xsl:if test="@units='p' and @end">
248             <xsl:text>p.</xsl:text>
249         </xsl:if>
250         <xsl:text>\thinspace </xsl:text>
251         <xsl:value-of select="@begin"/>
252         <xsl:if test="@end">
253             <xsl:text>--</xsl:text>
254             <xsl:value-of select="@end"/>
255         </xsl:if>
256         <xsl:if test="@annotations">
257             <xsl:if test="@units and @begin">
258                 <xsl:text>,</xsl:text>
259             </xsl:if>
260             <xsl:value-of
261                 select="normalize-space(
262                     @annotations)"/>
263             </xsl:if>
264             <xsl:text>]</xsl:text>
265         </xsl:if>
266         <!-- floating URI afterwards -->
267         <xsl:if test="@xlink:href">
268             <xsl:text>[\protect\url{</xsl:text>
269             <xsl:value-of
270                 select="replace(@xlink:href,'#','\\#')"/>
271             <xsl:text>}]</xsl:text>
272         </xsl:if>

```

`citelink` The final part is the actual identity token.

```

273     <xsl:text>{</xsl:text>
274     <xsl:value-of
275         select="translate(
276             normalize-space(@linkend),
277             ' ',',')"/>

```

```

278         <xsl:text>}</xsl:text>
279     </xsl:otherwise>
280 </xsl:choose>
281 </xsl:otherwise>
282 </xsl:choose>
283 <xsl:if
284     test="(//db:bibliography[@arch='biblatex']
285         or
286         //db:bibliolist[@arch='biblatex'])
287         and @xrefstyle='field' and @remap">
288     <xsl:text>{</xsl:text>
289     <xsl:value-of select="@remap"/>
290     <xsl:text>}</xsl:text>
291     <xsl:if test="contains(@remap,'title')">
292         <xsl:text>}</xsl:text>
293     </xsl:if>
294 </xsl:if>
295 </xsl:template>

```

bibliolist A `<bibliolist>` can occur elsewhere in the document, so we omit it here; and the exclusion for `<section>` is for the warnings in RFC2119.

```

296 <xsl:template match="db:bibliolist
297                 [parent::db:section]"/>

```

bibliography The bibliography container may have a title, so use it.

```

298 <xsl:template
299     match="db:bibliography |
300           db:bibliolist
301           [not(ancestor::db:bibliography)
302             and
303             not(parent::db:section)]">
304     <xsl:if test="db:title">
305         <xsl:value-of select="$armour"/>
306         <xsl:text>\renewcommand{\bibname}{</xsl:text>
307         <xsl:value-of select="db:title"/>
308         <xsl:text>}&#xa;</xsl:text>
309     </xsl:if>

```

biblioallcite If there are some `<exceptionname>`s but no direct references to RFC2119, *or* if there are no actual citations (`<biblioref>`, `<citetitle>`, or `<blockquote>`), then list everything provided.

```

310 <xsl:if
311     test="(//db:exceptionname
312         and not(//db:biblioref
313             [@linkend='rfc2119']))
314         or
315         (count(//db:biblioref) +
316         count(//db:citetitle[@linkend]) +
317         count(//db:blockquote[@linkend])=0)">
318     <xsl:value-of select="$armour"/>

```

```

319     <xsl:text>\nocite{*}&#xa;</xsl:text>
320 </xsl:if>

```

biblionewpage When called from `db2dtx.xsl`, start a new page and set the right-hand margin ragged (based on `$armour` being non-null in *ClassPack*).

```

321     <xsl:if test="$armour!=''">
322         <xsl:value-of select="$armour"/>
323         <xsl:text>\clearpage&#xa;</xsl:text>
324         <xsl:value-of select="$armour"/>
325         <xsl:text>\raggedright&#xa;</xsl:text>
326     </xsl:if>

```

bibliofile Set the name for the output BIB_TE_X file: either specified in `@xlink:href` or inherited from the `$name` variable.

```

327     <xsl:variable name="file">
328         <xsl:choose>
329             <xsl:when test="@xlink:href">
330                 <xsl:value-of select="@xlink:href"/>
331             </xsl:when>
332             <xsl:otherwise>
333                 <xsl:value-of select="$name"/>
334                 <xsl:text>.bib</xsl:text>
335             </xsl:otherwise>
336         </xsl:choose>
337     </xsl:variable>

```

biblioprintbiblatex The default is to assume biblatex is used: any other biblatex specifications MUST be given in the Preamble. The `@remap` attribute may hold the name of an alternative `\printbibliography` header format defined elsewhere, in which case use the `\RaggedRight` from `ragged2e`; otherwise just use normal `\raggedright`.

```

338     <xsl:choose>
339         <xsl:when test="local-name()='bibliography' and
340             (@arch='biblatex' or not(@arch))">
341             <xsl:value-of select="$armour"/>
342             <xsl:choose>
343                 <!-- @remap can specify an alternate style like shortbib -->
344                 <xsl:when test="@remap">
345                     <xsl:text>\RaggedRight</xsl:text>
346                     <xsl:text>\markboth{\refname}{\refname}</xsl:text>
347                 </xsl:when>
348                 <xsl:otherwise>
349                     <xsl:text>\raggedright</xsl:text>
350                 </xsl:otherwise>
351             </xsl:choose>
352             <xsl:text>\printbibliography</xsl:text>
353             <xsl:if test="@remap">
354                 <xsl:text>[heading=</xsl:text>
355                 <xsl:value-of select="@remap"/>
356                 <xsl:text>]</xsl:text>

```

```

357         </xsl:if>
358         <xsl:text>\clearpage&#xa;</xsl:text>
359     </xsl:when>

```

`biblioprintbibtex` For old BibTeX support (soon to be deprecated in *ClassPack*), construct the `\bibliography` and `\bibliographystyle` commands. The default style is APA.

```

360     <xsl:when
361         test="local-name()='bibliography'
362             and @arch='bibtex'">
363         <xsl:value-of select="$armour"/>
364         <xsl:text>\bibliography{</xsl:text>
365         <xsl:value-of
366             select="substring-before($file, '.bib')"/>
367         <xsl:text>}&#xa;</xsl:text>
368         <xsl:value-of select="$armour"/>
369         <xsl:text>\bibliographystyle{</xsl:text>
370         <xsl:choose>
371             <xsl:when test="@label">
372                 <xsl:value-of select="@label"/>
373             </xsl:when>
374             <xsl:otherwise>
375                 <xsl:text>apacite</xsl:text>
376             </xsl:otherwise>
377         </xsl:choose>
378         <xsl:text>}&#xa;</xsl:text>
379     </xsl:when>
380 </xsl:choose>

```

`bibliodata` Output the bibliography in BibTeX format, grouping the entries by type for convenience, and sorting by surname of first author.

Added inclusion of RFC2119.

```

381     <xsl:result-document format="textFormat"
382         href="{ $file}">
383         <xsl:for-each-group group-by="@xreflabel"
384             select="db:biblioentry">
385             <xsl:sort select="current-grouping-key()"/>
386             <xsl:text>&#xa;%% </xsl:text>
387             <xsl:value-of
388                 select="upper-case(current-grouping-key())"/>
389             <xsl:text>&#xa;</xsl:text>
390             <xsl:apply-templates select="current-group()">
391                 <xsl:sort
392                     select="descendant::db:author[1]
393                         /db:personname/db:surname
394                         |
395                         descendant::db:author[1]
396                         /db:orgname"/>
397                 <xsl:sort select="db:date/@YYYY-MM-DD"/>
398             </xsl:apply-templates>
399         </xsl:for-each-group>
400         <xsl:if test="//db:exceptionname

```

```

401             and not(db:biblioentry[@xml:id='rfc2119'])">
402     <xsl:text>%% Added automatically</xsl:text>
403     <xsl:apply-templates
404       select="document('rfc2119.xml')
405               /db:section/db:bibliolist
406               /db:biblioentry"/>
407   </xsl:if>
408 </xsl:result-document>
409 </xsl:template>
410
411 <xsl:template match="db:bibliography/db:title"/>
412

```

The title has already been dealt with above, so ignore it here.

12.1.2 Bibliographic entries

All fields are output in the standard BibTeX format.

biblioentry Some interventions are needed to accommodate standards, patents, and web pages. After the field data, there is scope to use the content of the `@role` attribute for the publisher of a standard, something not otherwise provided for. Similarly, for an article in a Proceedings where the publisher is not given, add the value 'Unpublished'. Also, if a date is not given, but conference dates *are*, try to detect the 'right' date.

```

413 <xsl:template match="db:biblioentry">
414   <xsl:text>&#xa;@</xsl:text>
415   <xsl:choose>
416     <xsl:when test="@xreflabel='standard' and @role">
417       <xsl:text>techreport</xsl:text>
418     </xsl:when>
419     <xsl:when test="@xreflabel='patent'
420                   or
421                   @xreflabel='webpage'">
422       <xsl:text>misc</xsl:text>
423     </xsl:when>
424     <xsl:otherwise>
425       <xsl:value-of select="@xreflabel"/>
426     </xsl:otherwise>
427   </xsl:choose>
428   <xsl:text>{</xsl:text>
429   <xsl:value-of select="@xml:id"/>
430   <xsl:apply-templates/>
431   <xsl:if test="@xreflabel='standard' and @role">
432     <xsl:text>,&#xa; publisher &#x9; = {</xsl:text>
433     <xsl:value-of select="@role"/>
434     <xsl:text>}</xsl:text>
435   </xsl:if>
436   <xsl:if test="(@xreflabel='inproceedings'
437                 or
438                 @xreflabel='inreference')
439                 and not(descendant::db:publisher)
440                 and not(descendant::db:publishername)

```

```

441         and not(descendant::db:orgname)
442         and not(descendant::db:volumenum)
443         and not(descendant::db:issuenum)"/>
444     <xsl:text>,&#xa; publisher &#x9; = {Unpublished}</xsl:text>
445 </xsl:if>
446 <xsl:if test="not(descendant::db:date)
447         and not(descendant::db:confnum[number(.)>999])
448         and descendant::db:confdates">
449     <xsl:for-each select="descendant::db:confdates[1]">
450         <xsl:call-template name="dodate"/>
451     </xsl:for-each>
452 </xsl:if>
453 <xsl:text>&#xa;}&#xa;</xsl:text>
454 </xsl:template>

```

bibliomisc If the `bibliomisc` element type is used, make it the reference type.

```

455 <xsl:template match="db:bibliomisc">
456     <xsl:text>,&#xa; type &#x9; = {</xsl:text>
457     <xsl:apply-templates/>
458     <xsl:text></xsl:text>
459 </xsl:template>

```

bibliosets The `<biblioset>` element keeps article- or chapter-specific data separate from that of the enclosing journal, book, etc. In this case, a special treatment is needed for facsimiles and topic-chapters in books.

bibliosets **TODO: Use of bibliosets needs rewriting**

```

460 <xsl:template match="db:biblioset | db:bibliomset">
461     <xsl:choose>
462         <xsl:when test="@relation='facsimile'">
463             <xsl:text>,&#xa; note &#x9; = {</xsl:text>
464             <xsl:text>Facsimile edition </xsl:text>
465             <xsl:if test="db:title[@role='translation']">
466                 <xsl:text>`</xsl:text>
467                 <xsl:value-of select="db:title"/>
468                 <xsl:text>'</xsl:text>
469             </xsl:if>
470             <xsl:if test="db:publisher">
471                 <xsl:text>,</xsl:text>
472                 <xsl:value-of
473                 select="db:publisher/db:publishername"/>
474                 <xsl:text>,</xsl:text>
475                 <xsl:value-of select="db:publisher/db:address"/>
476             </xsl:if>
477             <xsl:if test="db:date">
478                 <xsl:text>,</xsl:text>
479                 <xsl:value-of select="db:date"/>
480             </xsl:if>
481             <xsl:if test="db:address">
482                 <xsl:text>,</xsl:text>

```

```

483         <xsl:value-of select="db:address"/>
484     </xsl:if>
485     <xsl:text>.</xsl:text>
486 </xsl:when>
487 <xsl:otherwise>
488     <xsl:if test="@relation='inbook' and @remap">
489         <xsl:text>,&#xa; chapter &#x9; = {</xsl:text>
490         <xsl:value-of select="@remap"/>
491         <xsl:text>}</xsl:text>
492     </xsl:if>
493     <xsl:apply-templates/>
494 </xsl:otherwise>
495 </xsl:choose>
496 </xsl:template>

```

Authors

`biblioauthgroup` For multiple authors, write the names first, then try to group affiliations so as not to duplicate them.

`biblioauthgroup` **TODO: Rewrite this using grouping**

```

497 <xsl:template match="db:biblioentry//db:authorgroup">
498     <xsl:apply-templates/>
499     <xsl:if test="(db:author|db:editor)//db:affiliation">
500         <xsl:text>,&#xa; affiliation &#x9; = {</xsl:text>
501         <xsl:for-each
502             select="(db:author|db:editor)//db:affiliation">
503             <xsl:variable name="bibentry"
504                 select="generate-id(ancestor::db:biblioentry)"/>
505             <xsl:if
506                 test="count(preceding::db:affiliation
507                     [generate-id(ancestor::db:biblioentry)
508                     =$bibentry]) > 0">
509                 <xsl:text> and </xsl:text>
510             </xsl:if>
511             <xsl:if test="db:orgdiv">
512                 <xsl:apply-templates select="db:orgdiv/node()"/>
513                 <xsl:text>, </xsl:text>
514             </xsl:if>
515             <xsl:apply-templates select="db:orgname/node()"/>
516             <xsl:if test="db:address">
517                 <xsl:text>, </xsl:text>
518                 <xsl:apply-templates select="db:address/node()"/>
519             </xsl:if>
520         </xsl:for-each>
521         <xsl:text>}</xsl:text>
522     </xsl:if>
523 </xsl:template>

```

`biblioauthed` For authors and editors, start by detecting the first of its type and punctuate accordingly.

```

524 <xsl:template
525     match="db:biblioentry//db:author |
526           db:biblioentry//db:editor |
527           db:biblioentry//db:authorgroup/db:author |
528           db:biblioentry//db:authorgroup/db:editor">
529     <!-- label if first -->
530     <xsl:if
531         test="count(preceding-sibling::*
532                     [name()=current()/name()])=0">
533         <xsl:text>,&#xa; </xsl:text>
534         <xsl:value-of select="name()"/>
535         <xsl:text> &#x9; = {</xsl:text>
536     </xsl:if>
537     <!-- separator otherwise -->
538     <xsl:if
539         test="preceding-sibling::*[name()=current()/name()]">
540         <xsl:text> and </xsl:text>
541     </xsl:if>

```

biblioauthsolo Do each person one at a time: first name first, then any other name, then surname. Organisations as authors get tested for acronym status. Names containing spaces, or signalled by the @role attribute, get additional curly braces for protection against capitalising routines in BibTeX.

```

542 <!-- each person, if there are any (maybe orgname) -->
543 <xsl:for-each select="db:personname">
544     <!-- normal first name -->
545     <xsl:if test="db:firstname">
546         <xsl:choose>
547             <xsl:when test="db:firstname/@role='as-is'">
548                 <xsl:text>{</xsl:text>
549                 <xsl:value-of select="db:firstname"/>
550                 <xsl:text>}</xsl:text>
551             </xsl:when>
552             <xsl:otherwise>
553                 <xsl:value-of select="db:firstname"/>
554             </xsl:otherwise>
555         </xsl:choose>
556     </xsl:if>
557     <!-- optional other name -->
558     <xsl:if test="db:othername">
559         <xsl:text> {</xsl:text>
560         <xsl:value-of select="db:othername"/>
561         <xsl:text>}</xsl:text>
562     </xsl:if>
563     <!-- special handling of spaces and punctuation in surnames -->
564     <xsl:choose>
565         <xsl:when
566             test="contains(db:surname,' ')
567                 or
568                 contains(db:surname,' ')">
569             <xsl:text> {</xsl:text>
570             <xsl:value-of select="db:surname"/>

```

```

571         <xsl:text>}</xsl:text>
572     </xsl:when>
573     <xsl:otherwise>
574         <xsl:text> </xsl:text>
575         <xsl:value-of select="db:surname"/>
576     </xsl:otherwise>
577 </xsl:choose>
578 <xsl:if test="db:honorific">
579     <xsl:text> </xsl:text>
580     <xsl:value-of select="db:honorific"/>
581 </xsl:if>
582 </xsl:for-each>
583 <xsl:choose>
584     <!-- institutional organisation name -->
585     <xsl:when test="db:orgname">
586         <xsl:text>{</xsl:text>
587         <xsl:choose>
588             <!-- acronym -->
589             <xsl:when test=".=upper-case(.)">
590                 <xsl:text>\acro{</xsl:text>
591                 <xsl:apply-templates/>
592                 <xsl:text>}</xsl:text>
593             </xsl:when>
594             <xsl:otherwise>
595                 <xsl:apply-templates/>
596             </xsl:otherwise>
597         </xsl:choose>
598         <xsl:if test="db:orgdiv">
599             <xsl:text> {</xsl:text>
600             <xsl:apply-templates
601                 select="db:orgdiv/node()"/>
602             <xsl:text>}</xsl:text>
603         </xsl:if>
604         <xsl:text>}</xsl:text>
605     </xsl:when>
606     <!-- just text? -->
607     <xsl:otherwise>
608         <xsl:text> </xsl:text>
609     </xsl:otherwise>
610 </xsl:choose>

```

`biblioauthterm` Terminate the authors with the curly brace. Affiliations have already been dealt with.

```

611     <xsl:if test="count(following-sibling::*
612                         [name()='current()/name()])=0">
613         <xsl:text>}</xsl:text>
614     </xsl:if>
615 </xsl:template>
616
617 <xsl:template match="db:biblioentry//db:affiliation"/>

```

`bibliocorpauth` Handle corporate authors the same as human authors, and provide a short au-

thor field if there is a value in @remap.

```

618 <xsl:template match="db:biblioentry//db:corpauthor">
619   <xsl:if
620     test="count(preceding-sibling::db:author |
621               preceding-sibling::db:corpauthor)=0">
622     <xsl:text>,&#xa; author &#x9; = {{</xsl:text>
623   </xsl:if>
624   <xsl:if test="preceding-sibling::db:author |
625               preceding-sibling::db:corpauthor">
626     <xsl:text> and </xsl:text>
627   </xsl:if>
628   <xsl:choose>
629     <xsl:when test="@remap">
630       <xsl:value-of select="@remap"/>
631     </xsl:when>
632     <xsl:when test="not(@remap)">
633       <xsl:text>{{</xsl:text>
634       <xsl:apply-templates/>
635       <xsl:text>}}</xsl:text>
636     </xsl:when>
637     <xsl:otherwise>
638       <xsl:apply-templates/>
639     </xsl:otherwise>
640   </xsl:choose>
641   <xsl:if
642     test="count(following-sibling::db:author |
643               following-sibling::db:corpauthor)=0">
644     <xsl:text>}}</xsl:text>
645     <xsl:if
646       test="@remap and
647             count(preceding-sibling::db:author |
648                   preceding-sibling::db:corpauthor)=0">
649       <xsl:text>,&#xa; shortauthor &#x9; = {{</xsl:text>
650       <xsl:value-of select="@remap"/>
651       <xsl:text>}}</xsl:text>
652     </xsl:if>
653   </xsl:if>
654 </xsl:template>

```

biblioorgs Organisations map to institution and can have divisions (like a subtitle).

```

655 <xsl:template match="db:org">
656   <xsl:apply-templates/>
657 </xsl:template>
658
659 <xsl:template match="db:org/db:orgname">
660   <xsl:text>,&#xa; institution &#x9; = {{</xsl:text>
661   <xsl:apply-templates/>
662   <xsl:if test="following-sibling::db:orgdiv">
663     <xsl:text>, </xsl:text>
664     <xsl:apply-templates
665       select="following-sibling::db:orgdiv/node()"/>
666   </xsl:if>

```

```

667     <xsl:text>}}</xsl:text>
668 </xsl:template>
669
670 <xsl:template match="db:org/db:orgdiv | db:author/db:orgdiv"/>

```

bibliotitles This template tries to deal with all titles anywhere, using position to dictate the field name. A `@role` can override the field name; article titles are inside their own `<biblioset>`; a title in an article entry which is *outside* a `<biblioset>` is the name of the journal...

bibliotitles **TODO: This needs to use biblioset for journals**

...books, collections, and proceedings similarly use `booktitle`. Titles MAY contain markup, so this template distinguishes. That deals with naming and content,

```

671 <xsl:template match="db:biblioentry//db:title">
672   <xsl:text>,&#xa; </xsl:text>
673   <xsl:choose>
674     <xsl:when test="@role">
675       <xsl:value-of select="@role"/>
676     </xsl:when>
677     <xsl:when
678       test="parent::db:biblioset[@relation='article']">
679       <xsl:text>title</xsl:text>
680     </xsl:when>
681     <xsl:when
682       test="parent::db:biblioentry/@xreflabel='article'
683         or
684         parent::db:biblioset[@relation='journal']">
685       <xsl:text>journaltitle</xsl:text>
686     </xsl:when>
687     <xsl:when
688       test="(parent::db:biblioset/@relation='book' or
689         parent::db:biblioset/@relation='reference')
690         and (
691           ancestor::db:biblioentry/
692             @xreflabel='incollection'
693         or
694           ancestor::db:biblioentry/
695             @xreflabel='inproceedings'
696         or
697           ancestor::db:biblioentry/
698             @xreflabel='inreference')">
699       <xsl:text>booktitle</xsl:text>
700     </xsl:when>
701     <xsl:when
702       test="parent::db:biblioentry and
703         parent::db:biblioentry/db:biblioset
704         [@relation='article']">
705       <xsl:text>booktitle</xsl:text>
706     </xsl:when>
707     <xsl:otherwise>
708       <xsl:text>title</xsl:text>

```

```

709     </xsl:otherwise>
710 </xsl:choose>
711 <xsl:text> &#x9; = {</xsl:text>
712 <xsl:choose>
713   <xsl:when test="count(*)=0">
714     <xsl:value-of select="normalize-space(.)"/>
715   </xsl:when>
716   <xsl:otherwise>
717     <xsl:apply-templates/>
718   </xsl:otherwise>
719 </xsl:choose>

```

`bibliosubtitles` If the application is using [old] Bib_T_EX (not biblatex) — as evidenced by the `@label` set to "apacite" — append any subtitle separated by a colon. This is non-relevant when using biblatex which has its own subtitle field.

```

720   <xsl:if test="ancestor::db:bibliography
721     /@label='apacite'">
722     <xsl:for-each
723       select="following-sibling::db:subtitle
724         [@role=current()/@role or
725         (not(@role) and not(current()/@role))]">
726       <xsl:choose>
727         <xsl:when test="starts-with(.,'(')">
728           <xsl:text> </xsl:text>
729         </xsl:when>
730         <xsl:when test="@remap">
731           <xsl:text> \emph{</xsl:text>
732           <xsl:value-of select="@remap"/>
733           <xsl:text>} </xsl:text>
734         </xsl:when>
735         <xsl:otherwise>
736           <xsl:text>: </xsl:text>
737         </xsl:otherwise>
738       </xsl:choose>
739       <xsl:text>&#xa; &#x9; </xsl:text>
740       <xsl:choose>
741         <xsl:when test="count(*)=0">
742           <xsl:value-of select="normalize-space(.)"/>
743         </xsl:when>
744         <xsl:otherwise>
745           <xsl:apply-templates/>
746         </xsl:otherwise>
747       </xsl:choose>
748     </xsl:for-each>
749   </xsl:if>
750   <xsl:text>}}</xsl:text>
751 </xsl:template>
752
753 <xsl:template
754   match="db:biblioentry//db:subtitle
755     [ancestor::db:bibliography
756     /@label='apacite']"/>
757

```

```

758 <xsl:template
759     match="db:biblioentry//db:subtitle
760         [not(ancestor::db:bibliography
761             /@label='apacite')]">
762     <xsl:choose>
763         <xsl:when
764             test="parent::db:biblioset
765                 [@relation='book'
766                 or
767                 @relation='article']">
768             <xsl:text>,&#xa; subtitle &#x9; = {{</xsl:text>
769         </xsl:when>
770         <xsl:when
771             test="parent::db:biblioset
772                 [@relation='journal']">
773             <xsl:text>,&#xa; journalsubtitle &#x9; = {{</xsl:text>
774         </xsl:when>
775         <xsl:when
776             test="parent::db:biblioentry
777                 [@xreflabel='inbook'
778                 or
779                 @xreflabel='incollection'
780                 or
781                 @xreflabel='inreference']
782             and
783             parent::db:biblioentry/db:biblioset
784                 [@relation='article']">
785             <xsl:text>,&#xa; booksubtitle &#x9; = {{</xsl:text>
786         </xsl:when>
787         <xsl:otherwise>
788             <xsl:text>,&#xa; subtitle &#x9; = {{</xsl:text>
789         </xsl:otherwise>
790     </xsl:choose>
791     <xsl:apply-templates/>
792     <xsl:text>}}</xsl:text>
793 </xsl:template>

```

`biblioshorttitles` Known short titles are retained (biblatex only; ignored by BibTeX).

```

794 <xsl:template match="db:titleabbrev">
795     <xsl:text>,&#xa; shorttitle &#x9; = {{</xsl:text>
796     <xsl:apply-templates/>
797     <xsl:text>}}</xsl:text>
798 </xsl:template>

```

`bibliosources` [`db:biblioid` and `db:isbn` and `db:issn`]

Support for source references (ISBN, ISSN, DOI, etc) differs between BibTeX and biblatex. First, set the fieldname according to the way it is encoded; then add extra curly braces for a titleaddon to prevent it being interfered with; then add any prefix needed (eg LoC); add a `\url` command if it's a URI in a different field; and finally add the date accessed, if any.

```

799 <xsl:template
800     match="db:biblioid | db:isbn | db:issn">
801     <xsl:text>,&#xa; </xsl:text>
802     <!-- fieldname -->
803     <xsl:choose>
804         <xsl:when test="@class='uri'">
805             <xsl:text>url</xsl:text>
806         </xsl:when>
807         <xsl:when test="@class='doi'">
808             <xsl:value-of select="@class"/>
809         </xsl:when>
810         <xsl:when test="@class='pubsnumber'">
811             <xsl:text>titleaddon</xsl:text>
812         </xsl:when>
813         <xsl:when test="@class='other'">
814             <xsl:text>titleaddon</xsl:text>
815         </xsl:when>
816         <xsl:when
817             test="local-name()='isbn' or
818                 @class='isbn' or
819                 ancestor::db:biblioentry/@xreflabel='book'">
820             <xsl:text>isbn</xsl:text>
821         </xsl:when>
822         <xsl:when
823             test="local-name()='issn' or
824                 @class='issn' or
825                 ancestor::db:biblioentry/@xreflabel='article'">
826             <xsl:text>issn</xsl:text>
827         </xsl:when>
828         <xsl:otherwise>
829             <xsl:text>number</xsl:text>
830         </xsl:otherwise>
831     </xsl:choose>
832     <xsl:text> &#x9; = {</xsl:text>
833     <!-- shield -->
834     <xsl:if test="@class='pubsnumber'">
835         <xsl:text>{</xsl:text>
836     </xsl:if>
837     <!-- contents -->
838     <xsl:choose>
839         <xsl:when test="@class='other' and @otherclass">
840             <xsl:value-of select="@otherclass"/>
841             <xsl:text> </xsl:text>
842         </xsl:when>
843         <xsl:when test="@class='libraryofcongress'">
844             <xsl:text>LoC</xsl:text>
845             <xsl:text>: </xsl:text>
846         </xsl:when>
847     </xsl:choose>
848     <!-- if content is a URI but it's not a URI field -->
849     <xsl:if
850         test="matches(.,'^http[s]?://')
851             and
852             not(@class='uri')">

```

```

853     <xsl:text>\url{</xsl:text>
854 </xsl:if>
855 <!-- now the actual value -->
856 <xsl:choose>
857     <xsl:when test="@xlink:href">
858         <xsl:value-of select="@xlink:href"/>
859     </xsl:when>
860     <xsl:otherwise>
861         <xsl:value-of select="."/>
862     </xsl:otherwise>
863 </xsl:choose>
864 <xsl:if
865     test="matches(.,'^http[s]?://')
866         and
867         not(@class='uri')">
868     <xsl:text></xsl:text>
869 </xsl:if>
870 <!-- shield -->
871 <xsl:if test="@class='pubsnumber'">
872     <xsl:text></xsl:text>
873 </xsl:if>
874 <xsl:text></xsl:text>
875 <!-- generate additional fields: urldate -->
876 <xsl:choose>
877     <xsl:when
878         test="@YYYY-MM-DD
879             and
880             (ancestor::db:bibliography/@label='apacite'
881             or
882             ancestor::db:bibliography/@label='apa')">
883     <xsl:choose>
884         <xsl:when
885             test="ancestor::db:bibliography/@label='apacite'">
886             <xsl:text>,&#xa; lastchecked &#x9; = {</xsl:text>
887             <xsl:value-of
888                 select="format-date(@YYYY-MM-DD,'[D] [Mn] [Y]')"/>
889             </xsl:when>
890             <xsl:otherwise>
891                 <xsl:text>,&#xa; urldate &#x9; = {</xsl:text>
892                 <xsl:value-of select="@YYYY-MM-DD"/>
893             </xsl:otherwise>
894         </xsl:choose>
895     <xsl:text></xsl:text>
896 </xsl:when>
897 <xsl:when
898     test="@YYYY-MM-DD
899         and
900         not(ancestor::db:biblioentry//db:date)">
901     <xsl:text>,&#xa; year &#x9; = {</xsl:text>
902     <xsl:value-of
903         select="format-date(@YYYY-MM-DD,'[Y]')"/>
904     <xsl:text></xsl:text>
905     <xsl:if test="string-length(@YYYY-MM-DD)>5">
906         <xsl:text>,&#xa; month &#x9; = {</xsl:text>

```

```

907         <!-- month with MNn is wrong for biber -->
908         <xsl:value-of
909           select="format-date(@YYYY-MM-DD,'[M]')"/>
910         <xsl:text></xsl:text>
911       </xsl:if>
912     </xsl:when>
913   </xsl:choose>
914   <!-- note when it's a Usenet newsgroup -->
915   <xsl:if test="matches(.,'^news:')">
916     <xsl:text>,&#xa; note &#x9; = {</xsl:text>
917     <xsl:text>In Usenet newsgroup \texttt{</xsl:text>
918     <xsl:value-of
919       select="replace(substring-after(.,'news:')
920         ,'^[a-z\.\.]*$', '')"/>
921     <xsl:text>}, also available from Google Groups</xsl:text>
922     <xsl:text></xsl:text>
923   </xsl:if>
924 </xsl:template>

```

Publishers

`publisher` For the misc document type, use the `howpublished` field; otherwise pass through to the content elements.

```

925 <xsl:template match="db:publisher">
926   <xsl:choose>
927     <xsl:when
928       test="ancestor::db:biblioentry/@xreflabel='misc'">
929       <xsl:text>,&#xa; howpublished &#x9; = {</xsl:text>
930       <xsl:apply-templates select="db:address/node()"/>
931       <xsl:text>: </xsl:text>
932       <xsl:apply-templates
933         select="db:publishername/node()"/>
934       <xsl:text></xsl:text>
935     </xsl:when>
936     <xsl:otherwise>
937       <xsl:apply-templates/>
938     </xsl:otherwise>
939   </xsl:choose>
940 </xsl:template>

```

`publishername` Again, set the field names for exceptions first.

```

941 <xsl:template match="db:publishername">
942   <xsl:text>,&#xa; </xsl:text>
943   <xsl:choose>
944     <xsl:when
945       test="ancestor::db:biblioentry/
946         @xreflabel='techreport'
947       or
948         ancestor::db:biblioentry/
949         @xreflabel='standard'
950       or
951         ancestor::db:biblioentry/

```

```

952         @xreflabel='manual'"">
953         <xsl:text>organization</xsl:text>
954     </xsl:when>
955     <xsl:when
956         test="ancestor::db:biblioentry/
957             @xreflabel='phdthesis'
958             or
959             ancestor::db:biblioentry/
960             @xreflabel='mastersthesis'"">
961         <xsl:text>institution</xsl:text>
962     </xsl:when>
963     <xsl:otherwise>
964         <xsl:text>publisher</xsl:text>
965     </xsl:otherwise>
966 </xsl:choose>
967 <xsl:text> &#x9; = {</xsl:text>
968 <xsl:apply-templates/>
969 <xsl:text>}</xsl:text>
970 </xsl:template>

```

pubaddress **Detect network addresses and handle separately.**

```

971 <xsl:template match="db:publisher/db:address">
972     <xsl:text>,&#xa; </xsl:text>
973     <xsl:choose>
974         <xsl:when
975             test="starts-with(normalize-space(.),'http://') or
976                 starts-with(normalize-space(.),'ftp://') or
977                 starts-with(normalize-space(.),'mailto:') or
978                 starts-with(normalize-space(.),'https://')">
979             <xsl:text>url</xsl:text>
980         </xsl:when>
981         <xsl:otherwise>
982             <xsl:text>address</xsl:text>
983         </xsl:otherwise>
984     </xsl:choose>
985     <xsl:text> &#x9; = {</xsl:text>
986     <xsl:apply-templates/>
987     <xsl:text>}</xsl:text>
988 </xsl:template>

```

version **Abuse the <releaseinfo> element to hold a software package version number.**

```

989 <xsl:template match="db:releaseinfo">
990     <xsl:text>,&#xa; version &#x9; = {</xsl:text>
991     <xsl:value-of select="."/>
992     <xsl:text>}</xsl:text>
993 </xsl:template>

```

Conferences

confgroup **Conference group data passes through but conference number is done separately within the conference title.**

```

994 <xsl:template match="db:confgroup">
995   <xsl:apply-templates/>
996 </xsl:template>
997
998 <xsl:template match="db:confnum"/>

```

conftitle Try to align conference numbers with titles that have the year embedded.

```

999 <xsl:template match="db:conftitle">
1000   <xsl:text>,&#xa; booktitle &#x9; = {{Proc. </xsl:text>
1001   <xsl:choose>
1002     <xsl:when test="../db:confnum">
1003       <xsl:choose>
1004         <xsl:when
1005           test="matches(.,'$apostrophe[0-9][0-9]$')">
1006           <xsl:value-of
1007             select="replace(.,
1008               '$apostrophe[0-9][0-9]$', '')"/>
1009         </xsl:when>
1010         <xsl:when
1011           test="matches(.,''[0-9][0-9]$')">
1012           <xsl:value-of
1013             select="replace(.,''[0-9][0-9]$', '')"/>
1014         </xsl:when>
1015         <xsl:when
1016           test="matches(.,'[0-9][0-9]$')">
1017           <xsl:value-of
1018             select="replace(.,'[0-9][0-9]$', '')"/>
1019         </xsl:when>
1020         <xsl:otherwise>
1021           <xsl:apply-templates/>
1022         </xsl:otherwise>
1023       </xsl:choose>
1024       <!-- now add the real year/number -->
1025       <xsl:choose>
1026         <xsl:when test="number(../db:confnum)>999">
1027           <xsl:text> </xsl:text>
1028           <xsl:value-of select="../db:confnum"/>
1029         </xsl:when>
1030         <xsl:otherwise>
1031           <xsl:apply-templates
1032             select="../db:confnum/node()"/>
1033           <xsl:text>: </xsl:text>
1034         </xsl:otherwise>
1035       </xsl:choose>
1036     </xsl:when>
1037     <!-- otherwise it's just the name of the conf -->
1038     <xsl:otherwise>
1039       <xsl:apply-templates/>
1040     </xsl:otherwise>
1041   </xsl:choose>
1042   <!-- if there are dates, use biblatex -->
1043   <xsl:text>}}</xsl:text>
1044 </xsl:template>

```

confsponsor The conference sponsor is the organisation.

```

1045 <xsl:template match="db:confsponsor">
1046   <xsl:text>,&#xa; organization &#x9; = {</xsl:text>
1047   <xsl:apply-templates/>
1048   <xsl:text>}</xsl:text>
1049 </xsl:template>

```

orguri Organisation URI.

```

1050 <xsl:template match="db:org/db:uri">
1051   <xsl:text>,&#xa; uri &#x9; = {</xsl:text>
1052   <xsl:apply-templates/>
1053   <xsl:text>}</xsl:text>
1054 </xsl:template>

```

conforgaddress Conference organiser address or location becomes the conference address.

```

1055 <xsl:template match="db:confgroup/db:address">
1056   <xsl:text>,&#xa; venue &#x9; = {</xsl:text>
1057   <xsl:apply-templates/>
1058   <xsl:text>}</xsl:text>
1059 </xsl:template>
1060 <xsl:template match="db:org/db:address">
1061   <xsl:text>,&#xa; location &#x9; = {</xsl:text>
1062   <xsl:apply-templates/>
1063   <xsl:text>}</xsl:text>
1064 </xsl:template>

```

confdates Real conference dates use the eventdate field.

```

1065 <xsl:template match="db:confdates">
1066   <xsl:text>,&#xa; eventdate &#x9; = {</xsl:text>
1067   <xsl:choose>
1068     <xsl:when test="@YYYY-MM-DD-from">
1069       <xsl:value-of select="@YYYY-MM-DD-from"/>
1070     <xsl:if test="@YYYY-MM-DD-to">
1071       <xsl:text></xsl:text>
1072       <xsl:value-of select="@YYYY-MM-DD-to"/>
1073     </xsl:if>
1074   </xsl:when>
1075   <xsl:otherwise>
1076     <xsl:value-of select="."/>
1077   </xsl:otherwise>
1078 </xsl:choose>
1079   <xsl:text>}</xsl:text>
1080 </xsl:template>

```

dates Original date field gets parsed separately.

```

1081 <xsl:template match="db:date">
1082   <xsl:call-template name="dodate"/>
1083 </xsl:template>

```

dodate Separate out year and month for the right fields.

```

1084 <xsl:template name="dodate">
1085   <xsl:text>,&#xa; year &#x9; = {</xsl:text>
1086   <xsl:choose>
1087     <xsl:when test="@YYYY-MM-DD!=''">
1088       <xsl:value-of
1089         select="format-date(xs:date(
1090           @YYYY-MM-DD),'[Y]')"/>
1091     </xsl:when>
1092     <xsl:when test="@YYYY-MM-DD-from!=''">
1093       <xsl:value-of
1094         select="format-date(xs:date(
1095           @YYYY-MM-DD-from),'[Y]')"/>
1096     </xsl:when>
1097     <xsl:otherwise>
1098       <xsl:apply-templates/>
1099     </xsl:otherwise>
1100   </xsl:choose>
1101   <xsl:text>}</xsl:text>
1102   <xsl:choose>
1103     <xsl:when test="@YYYY-MM-DD!=''">
1104       <xsl:text>,&#xa; month &#x9; = {</xsl:text>
1105       <xsl:value-of
1106         select="format-date(xs:date(
1107           @YYYY-MM-DD),'[M]')"/>
1108       <xsl:text>}</xsl:text>
1109     </xsl:when>
1110     <xsl:when test="@YYYY-MM-DD-from!=''">
1111       <xsl:text>,&#xa; month &#x9; = {</xsl:text>
1112       <xsl:value-of
1113         select="format-date(xs:date(
1114           @YYYY-MM-DD-from),'[M]')"/>
1115       <xsl:text>}</xsl:text>
1116     </xsl:when>
1117   </xsl:choose>
1118   <xsl:if test="ancestor::db:biblioentry/@xreflabel='pc'">
1119     <xsl:text>,&#xa; howpublished &#x9; = {</xsl:text>
1120     <xsl:text>Pers.\ Comm.</xsl:text>
1121   </xsl:if>
1122 </xsl:template>

```

pagenums Pages (range and count)

pagenums **TODO: Separate page count (books) from page range**

```

1123 <xsl:template match="db:pagenums | db:artpagenums">
1124   <xsl:choose>
1125     <xsl:when test="local-name()='pagenums'">
1126       <xsl:text>,&#xa; pagetotal &#x9; = {</xsl:text>
1127     </xsl:when>
1128     <xsl:when test="local-name()='artpagenums'">
1129       <xsl:text>,&#xa; pages &#x9; = {</xsl:text>

```

```

1130     </xsl:when>
1131 </xsl:choose>
1132 <xsl:choose>
1133     <!-- en-dash already present -->
1134     <xsl:when test="contains(.,'&#x2013;')">
1135         <xsl:value-of select="substring-before(.,'&#x2013;')"/>
1136         <xsl:text>--</xsl:text>
1137         <xsl:value-of select="substring-after(.,'&#x2013;')"/>
1138     </xsl:when>
1139     <!-- double-hyphen -->
1140     <xsl:when test="contains(.,'--')">
1141         <xsl:value-of select="."/>
1142     </xsl:when>
1143     <!-- single hyphen -->
1144     <xsl:when test="contains(.,'-'')">
1145         <xsl:value-of select="substring-before(.,'-'')"/>
1146         <xsl:text>--</xsl:text>
1147         <xsl:value-of select="substring-after(.,'-'')"/>
1148     </xsl:when>
1149     <xsl:otherwise>
1150         <xsl:value-of select="."/>
1151     </xsl:otherwise>
1152 </xsl:choose>
1153 <xsl:text>}</xsl:text>
1154 </xsl:template>

```

`volumenum` **Volume of a journal**

```

1155 <xsl:template match="db:volumenum">
1156     <xsl:choose>
1157         <xsl:when test="ancestor::db:biblioentry/@xreflabel='book'">
1158             <xsl:text>,&#xa; volumes &#x9; = {</xsl:text>
1159         </xsl:when>
1160         <xsl:otherwise>
1161             <xsl:text>,&#xa; volume &#x9; = {</xsl:text>
1162         </xsl:otherwise>
1163     </xsl:choose>
1164     <xsl:apply-templates/>
1165     <xsl:text>}</xsl:text>
1166 </xsl:template>

```

`issuenum` [`db:issuenum` and `db:productnumber`]

Issue number of a journal. The `<productnumber>` element can be used for conference numbers to avoid them being encoded as issues.

```

1167 <xsl:template match="db:issuenum | db:productnumber">
1168     <xsl:text>,&#xa; number &#x9; = {</xsl:text>
1169     <xsl:apply-templates/>
1170     <xsl:text>}</xsl:text>
1171 </xsl:template>

```

`edition` **Edition of a book**

edition **TODO: Recode to use ordinal**

```

1172 <xsl:template match="db:edition">
1173   <xsl:text>,&#xa; edition &#x9; = {</xsl:text>
1174   <xsl:choose>
1175     <xsl:when test="not(number())">
1176       <xsl:apply-templates/>
1177     </xsl:when>
1178     <xsl:otherwise>
1179       <xsl:apply-templates/>
1180     <xsl:choose>
1181       <xsl:when
1182         test="substring(.,string-length())='1' and
1183             substring(.,string-length()-1)!='11'">
1184         <xsl:text>st</xsl:text>
1185       </xsl:when>
1186       <xsl:when
1187         test="substring(.,string-length())='2' and
1188             substring(.,string-length()-1)!='12'">
1189         <xsl:text>nd</xsl:text>
1190       </xsl:when>
1191       <xsl:when
1192         test="substring(.,string-length())='3' and
1193             substring(.,string-length()-1)!='13'">
1194         <xsl:text>rd</xsl:text>
1195       </xsl:when>
1196       <xsl:otherwise>
1197         <xsl:text>th</xsl:text>
1198       </xsl:otherwise>
1199     </xsl:choose>
1200   </xsl:otherwise>
1201 </xsl:choose>
1202   <xsl:text>}</xsl:text>
1203 </xsl:template>

```

abstract **Abstracts in references are uncommon, but record one if it is provided.**

```

1204 <xsl:template match="db:abstract">
1205   <xsl:text>,&#xa; abstract &#x9; = {</xsl:text>
1206   <xsl:value-of select="normalize-space(.)"/>
1207   <xsl:text>}</xsl:text>
1208 </xsl:template>

```

address **An address which is not part of a publisher element, or a conference group, or an organisation is given the location field type.**

```

1209 <xsl:template
1210   match="db:address[
1211     not(parent::db:publisher)
1212     and not(parent::db:confgroup)
1213     and not(parent::db:org)]">
1214   <xsl:text>,&#xa; location &#x9; = {</xsl:text>
1215   <xsl:value-of select="normalize-space(.)"/>

```

```

1216     <xsl:text>}</xsl:text>
1217 </xsl:template>

```

orgname **Organisation names** (ie explicitly not publishers) are usually academic or research institutions.

```

1218 <xsl:template
1219     match="db:orgname[
1220         not(parent::db:author)
1221         and not(parent::db:org)]">
1222     <xsl:text>,&#xa; institution &#x9; = {</xsl:text>
1223     <xsl:value-of select="normalize-space(.)"/>
1224     <xsl:text>}</xsl:text>
1225 </xsl:template>

```

series **Series are just series.**

```

1226 <xsl:template match="db:seriesvolnums |
1227                 db:biblioset[@relation='series']">
1228     <xsl:text>,&#xa; series &#x9; = {</xsl:text>
1229     <xsl:value-of select="normalize-space(.)"/>
1230     <xsl:text>}</xsl:text>
1231 </xsl:template>

```

coverage **The bibliocoverage element type is used to express a part of a cited document like a chapter.**

```

1232 <xsl:template match="db:bibliocoverage">
1233     <xsl:text>,&#xa; chapter &#x9; = {</xsl:text>
1234     <xsl:value-of select="normalize-space(.)"/>
1235     <xsl:text>}</xsl:text>
1236 </xsl:template>

```

done **The End.**

```

1237 </xsl:stylesheet>

```

13 The figtab2latex.xsl script

figtabscript-comment Identify the file origin, date, time, etc.

```
1 <!-- The figtab2latex.xsl script (from the classpack package v.1.28, 2024-02-21) -->
2 <!-- Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16 -->
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

Figure and tables in documentation share some common features and are handled by code in a separate XSLT file included from db2dtx.xsl.

init The file starts with the XSLT declaration and the output type setting.

```
3 <xsl:stylesheet
4   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
5   xmlns:db="http://docbook.org/ns/docbook"
6   xmlns:xlink="http://www.w3.org/1999/xlink"
7   version="3.0">
8
9   <xsl:output method="text"/>
```

Tables

table shield Output a \LaTeX table environment, with each physical line shielded from Doc \TeX with the current value of the variable \$shield. Honour the float style, defaulting to **ht**. Tables are set in the current sans-serif type, one step-size smaller than the body text.

```
10 <xsl:template match="db:table">
11   <!-- number of columns -->
12   <xsl:variable name="ncols" select="db:tgroup/@cols"/>
13   <!-- work out total table width -->
14   <xsl:value-of select="$shield"/>
15   <xsl:text> \setlength{\CPKtabwid}{</xsl:text>
16   <xsl:call-template name="colwidths">
17     <xsl:with-param name="loc" select="db:tgroup"/>
18   </xsl:call-template>
19   <xsl:text> + </xsl:text>
20   <xsl:value-of select="$ncols - 1"/>
21   <xsl:text>\tabcolsep}&#xa;</xsl:text>
22   <!-- size and font control -->
23   <xsl:if test="@wordsize">
24     <xsl:value-of select="$shield"/>
25     <xsl:text> \fontsize{</xsl:text>
26     <xsl:value-of
27       select="tokenize(@wordsize, '/') [1]"/>
28     <xsl:text>}</xsl:text>
29     <xsl:value-of
30       select="tokenize(@wordsize, '/') [2]"/>
31     <xsl:text>}\selectfont</xsl:text>
32   </xsl:if>
```

```

33 <xsl:if test="@role">
34   <xsl:if test="not(@wordsize)">
35     <xsl:value-of select="$shield"/>
36     <xsl:text> </xsl:text>
37   </xsl:if>
38   <xsl:value-of select="@role"/>
39 </xsl:if>
40 <xsl:if test="@wordsize or @role">
41   <xsl:text>&#xa;</xsl:text>
42 </xsl:if>
43 <!-- start the table -->
44 <xsl:value-of select="$shield"/>
45 <xsl:text> \begin{</xsl:text>
46 <xsl:value-of select="@condition"/><!-- eg 'long' -->
47 <xsl:text>table}</xsl:text>
48 <xsl:choose>
49   <xsl:when test="@condition='long'">
50     <xsl:text></xsl:text>
51   </xsl:when>
52   <xsl:when test="@floatstyle">
53     <xsl:text>[</xsl:text>
54     <xsl:value-of select="@floatstyle"/>
55     <xsl:text>]</xsl:text>
56   </xsl:when>
57   <xsl:otherwise>
58     <xsl:text>[ht]</xsl:text>
59   </xsl:otherwise>
60 </xsl:choose>
61 <xsl:choose>
62   <!-- longtable has to do column parse BEFORE caption etc -->
63   <xsl:when test="@condition='long'">
64     <xsl:for-each select="db:tgroup">
65       <xsl:call-template name="colparse"/>
66       <xsl:apply-templates select="../db:title|../db:subtitle"/>
67       <xsl:apply-templates
68         select="
69           db:thead|
70           db:tbody|
71           db:tfoot
72         "/>
73     </xsl:for-each>
74   </xsl:when>
75   <!-- normal table -->
76   <xsl:otherwise>
77     <xsl:text>&#xa;</xsl:text>
78     <xsl:apply-templates/>
79   </xsl:otherwise>
80 </xsl:choose>
81 <!-- any prefatory test gets output as the last row of tabular
82   at the end of the tgroup template; longtables get it here -->
83 <xsl:if test="db:textobject[@role='footer'] and @condition='long'">
84   <xsl:value-of select="$shield"/>
85   <xsl:text> \multicolumn{</xsl:text>
86   <xsl:value-of select="$ncols"/>

```

```

87      <xsl:text>}>{\renewcommand{\baselinestretch}{0.8}}</xsl:text>
88      <xsl:text>\selectfont{\centering}\CPKprestrut</xsl:text>
89      <xsl:text>p{\CPKtabwid}}{\footnotesize&#xa;</xsl:text>
90      <xsl:apply-templates
91        select="db:textobject[@role='footer']/*"/>
92      <xsl:value-of select="$shield"/>
93      <xsl:text> }&#xa;</xsl:text>
94      <xsl:value-of select="$shield"/>
95      <xsl:text> \&#xa;</xsl:text>
96    </xsl:if>
97    <xsl:value-of select="$shield"/>
98    <xsl:text> \end{</xsl:text>
99    <xsl:value-of select="@condition"/><!-- eg 'long' -->
100   <xsl:text>table}&#xa;</xsl:text>
101   <!-- size and font control -->
102   <xsl:if test="@wordsize">
103     <xsl:value-of select="$shield"/>
104     <xsl:text> \normalsize</xsl:text>
105   </xsl:if>
106   <xsl:if test="@role">
107     <xsl:if test="not(@wordsize)">
108       <xsl:value-of select="$shield"/>
109       <xsl:text> </xsl:text>
110     </xsl:if>
111     <xsl:text>\normalfont</xsl:text>
112   </xsl:if>
113   <xsl:if test="@wordsize or @role">
114     <xsl:text>&#xa;</xsl:text>
115   </xsl:if>
116 </xsl:template>
117
118 <xsl:template match="db:table/db:textobject
119   [@role='footer']"/>

```

If there is a `<textobject>` element present, with a `@role` of "footer", use it *within the table environment* in smaller type, without adding any footnote number or symbol. This element will then be ignored when it is encountered in document order.

`tabletitle` Make the table `<title>` into the caption. A subtitle is not currently implemented. The `@xml:id` is used as the \LaTeX `\label`, if present.

```

120   <xsl:template match="db:table/db:title[1]
121     |
122     db:figure/db:title[1]">
123     <xsl:value-of select="$shield"/>
124     <xsl:text> \caption</xsl:text>
125     <!-- a second title is used for short caption -->
126     <xsl:if test="following-sibling::db:title">
127       <xsl:text>[</xsl:text>
128       <xsl:apply-templates
129         select="following-sibling::db:title/node()"/>
130       <xsl:text>]</xsl:text>
131     </xsl:if>

```

```

132     <xsl:text>{</xsl:text>
133     <xsl:apply-templates/>
134     <xsl:text>}</xsl:text>
135     <xsl:if test="../@xml:id
136             and
137             not(parent::db:table/@condition='long')">
138         <xsl:text>\label{</xsl:text>
139         <xsl:value-of select="../@xml:id"/>
140         <xsl:text>}</xsl:text>
141     </xsl:if>
142     <xsl:choose>
143         <xsl:when test="parent::db:table/@condition='long'">
144             <xsl:text>\\&#xa;</xsl:text>
145         </xsl:when>
146         <xsl:otherwise>
147             <xsl:text>\smallskip&#xa;</xsl:text>
148         </xsl:otherwise>
149     </xsl:choose>
150 </xsl:template>
151
152 <xsl:template
153     match="db:table/db:title[position()>1]
154         |
155         db:figure/db:title[position()>1]"/>

```

`informaltable` Informal tables are unnumbered and uncaptioned, in sans-serif type, and by default smaller size, but the `@wordsize` attribute can be used to specify a font size with two numbers separated by a slash, for the font size and baseline height, eg `"7/8"` for 7pt type on an 8pt baseline. The `@role` attribute can be used to hold specific L^AT_EX commands if needed. The content is a `<tgroup>` element, formatted the same way as a formal table.

```

156 <xsl:template match="db:informaltable">
157     <xsl:value-of select="$shield"/>
158     <xsl:text> \par\medskip{\sffamily</xsl:text>
159     <xsl:choose>
160         <xsl:when test="@role">
161             <xsl:value-of select="@role"/>
162         </xsl:when>
163         <xsl:when test="@wordsize">
164             <xsl:text>\fontsize{</xsl:text>
165             <xsl:value-of
166                 select="tokenize(@wordsize, '/') [1]"/>
167             <xsl:text>}{</xsl:text>
168             <xsl:value-of
169                 select="tokenize(@wordsize, '/') [2]"/>
170             <xsl:text>}\selectfont</xsl:text>
171         </xsl:when>
172         <xsl:otherwise>
173             <xsl:text>\small</xsl:text>
174         </xsl:otherwise>
175     </xsl:choose>
176     <xsl:if test="@xml:id">

```

```

177     <xsl:text>\label{</xsl:text>
178     <xsl:value-of select="@xml:id"/>
179     <xsl:text>}</xsl:text>
180   </xsl:if>
181   <xsl:text>&#xa;</xsl:text>
182   <xsl:apply-templates/>
183   <xsl:value-of select="$shield"/>
184   <xsl:text> }&#xa;</xsl:text>
185 </xsl:template>

```

`informaltableinfo` The `<info>` and `<title>` elements in an informal table can be used for a heading if required.

```

186 <xsl:template
187   match="db:informaltable/db:info">
188   <xsl:text>\begin{center}</xsl:text>
189   <xsl:apply-templates/>
190   <xsl:text>\end{center}</xsl:text>
191 </xsl:template>
192
193 <xsl:template
194   match="db:informaltable/db:info/db:title">
195   <xsl:text>\textbf{</xsl:text>
196   <xsl:apply-templates/>
197   <xsl:text>}</xsl:text>
198 </xsl:template>

```

`tgroup` The `<tgroup>` is the tabular component of formal and informal tables. It typically holds a set of `<colspec>` elements which specify the column types and widths, a `<thead>` for the column headings, a `<tbody>` for the body, and optionally a `<tfoot>` for totals.

`tabstart` The `tabstart` named template handles the column settings. If there is no `<thead>`, a line is drawn across all columns. The body and foot are processed in that order (*DocBook* specifies that the `<tfoot>` element occur earlier).

```

199 <xsl:template match="db:tgroup">
200   <!-- formal and informal tables get their tabular-start here;
201       longtable already did it -->
202   <xsl:if test="not(ancestor::db:table/@condition='long')">
203     <xsl:text>% \small\sffamily&#xa;</xsl:text>
204     <xsl:call-template name="tabstart"/>
205   </xsl:if>
206   <!-- process the head lines if any -->
207   <xsl:apply-templates select="db:thead"/>
208   <!-- if there wasn't one, draw a rule -->
209   <xsl:if test="not(db:thead)">
210     <xsl:value-of select="$shield"/>
211     <xsl:text> \hline&#xa;</xsl:text>
212   </xsl:if>
213   <!-- then do the body and the foot -->
214   <xsl:apply-templates select="db:tbody"/>
215   <xsl:apply-templates select="db:tfoot"/>

```

```

216 <!-- formal and informal tables get their tabular-end here -->
217 <xsl:if test="not(ancestor::db:table/@condition='long')">
218   <!-- any preparatory test gets output as the last row of longtable
219   in the table template; normal tables get it here -->
220   <xsl:if test="ancestor::db:table/db:textobject[@role='footer']
221     and not(ancestor::db:table/@condition='long')">
222     <xsl:value-of select="$shield"/>
223     <xsl:text> \multicolumn{</xsl:text>
224     <xsl:value-of select="count(db:colspec)"/>
225     <xsl:text>}}{\renewcommand{\baselinestretch}{0.8}</xsl:text>
226     <xsl:text>\selectfont{\centering}\CPKprestrut</xsl:text>
227     <xsl:text>p{\CPKtabwid}}{\footnotesize&#xa;</xsl:text>
228     <xsl:apply-templates
229       select="ancestor::db:table/db:textobject[@role='footer']/*"/>
230     <xsl:value-of select="$shield"/>
231     <xsl:text> }&#xa;</xsl:text>
232     <xsl:value-of select="$shield"/>
233     <xsl:text> \&#xa;</xsl:text>
234   </xsl:if>
235   <xsl:value-of select="$shield"/>
236   <xsl:text> \end{tabular}&#xa;</xsl:text>
237 </xsl:if>

```

tgroupfakefootnotes **WARNING: Experimental and unsupported option**

In document classes with a very large number of options (typically university theses, with options for subjects and types of degree), the tables describing them need extensive annotation. If the <tbody> has a @xlink:href attribute starting with the string "affil" and a matching set of <constraintdef> elements in the "data" part of the document, use them as values for a list of explanations.

```

238 <xsl:if
239   test="starts-with(db:tbody/@xlink:href,'affil')
240   and
241   count(//db:part[@xml:id='data']
242     //db:constraintdef
243     [@xml:id=concat(substring-before(
244       current()/db:tbody/@xlink:href,'-'),
245       'options')])
246     /db:methodsynopsis[tokenize(@arch,' ')=
247       substring-after(
248         current()/db:tbody/@xlink:href,'-')]
249     [db:modifier]]>0">
250   <xsl:value-of select="$shield"/>
251   <xsl:text> \scriptsize&#xa;</xsl:text>
252   <xsl:value-of select="$shield"/>
253   <xsl:text> \renewcommand{\labelenumi}{</xsl:text>
254   <xsl:text>\itshape\alph{enumi}\upshape}}&#xa;</xsl:text>
255   <xsl:text> \begin{enumerate}[noitemsep]&#xa;</xsl:text>
256   <xsl:for-each
257     select="//db:part[@xml:id='data']
258     //db:constraintdef[@xml:id=
259     concat(substring-before(
260       current()/db:tbody/@xlink:href,'-'),

```

```

261         'options'']]
262         /db:methodsynopsis[tokenize(@arch,' ')=
263         substring-after(
264         current()/db:tbody/@xlink:href,'-')]
265         /db:modifier">
266         <xsl:value-of select="$shield"/>
267         <xsl:text> \item The \texttt{</xsl:text>
268         <xsl:value-of select="../@xml:id"/>
269         <xsl:text>} option </xsl:text>
270         <xsl:apply-templates/>
271         <xsl:text>&#xa;</xsl:text>
272     </xsl:for-each>
273     <xsl:value-of select="$shield"/>
274     <xsl:text> \end{enumerate}&#xa;</xsl:text>
275     <xsl:value-of select="$shield"/>
276     <xsl:text> \setcounter{fnote}{0}&#xa;</xsl:text>
277 </xsl:if>

```

`tgroupend` Terminate the $\text{T}_{\text{E}}\text{X}$ group started earlier in the `tabstart` named template.

```

278     <xsl:value-of select="$shield"/>
279     <xsl:text> \par\endgroup&#xa;</xsl:text>
280 </xsl:template>

```

`tabstart` This template is called from within a `<tgrou>` to handle column settings. It starts a $\text{T}_{\text{E}}\text{X}$ group which is terminated at the end of the template for the `<tgrou>` element.

Any local settings in the form of $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ commands in the `@remap` attribute get honoured first. The tabular material is centred horizontally in the page, and the default shoulders are omitted unless the `@tgroustyle` is set to "overhang".

The `<colspec>` elements are handled in order by the three named templates `colprefix`, `colsettings`, and `colsuffix`, so the elements are ignored when they come up in document order.

`colprefix`
`colsettings`
`colsuffix`

```

281 <xsl:template name="tabstart">
282   <xsl:param name="loc">
283     <xsl:text></xsl:text>
284   </xsl:param>
285   <!-- called in the context of db:tgrou -->
286   <xsl:value-of select="$shield"/>
287   <xsl:text> \begingroup</xsl:text>
288   <xsl:value-of
289     select="parent::db:table/@remap |
290     parent::db:informaltable/@remap"/>
291   <xsl:text>&#xa;</xsl:text>
292   <xsl:value-of select="$shield"/>
293   <xsl:text> \centering&#xa;</xsl:text>
294   <xsl:value-of select="$shield"/>
295   <xsl:text> \begin{tabular}</xsl:text>
296   <xsl:call-template name="colparse"/>
297 </xsl:template>

```

```

298
299 <!-- colparse now separated out because of longtable support -->
300 <xsl:template name="colparse">
301   <xsl:param name="loc">
302     <xsl:text></xsl:text>
303   </xsl:param>
304   <!-- WARNING: if you edit this, don't forget to mirror the edits
305     in the colwidths template later on -->
306   <xsl:text>{</xsl:text>
307   <xsl:choose>
308     <xsl:when test="@tgroupstyle='overhang' or @colsep='1'">
309       <xsl:text>@{\quad}</xsl:text>
310     </xsl:when>
311     <xsl:otherwise>
312       <xsl:text>@{</xsl:text>
313     </xsl:otherwise>
314   </xsl:choose>
315   <xsl:value-of select="$shield"/>
316   <xsl:text>&#xa;</xsl:text>
317   <xsl:value-of select="$shield"/>
318   <!-- try to indent each column spec on its own line -->
319   <xsl:text>   </xsl:text>
320   <xsl:for-each select="db:colspec">
321     <xsl:choose>
322       <xsl:when test="@colsep='0'">
323         <xsl:text>@{\enspace}</xsl:text>
324       </xsl:when>
325       <xsl:when test="@colsep='1'">
326         <xsl:text>@{\qqquad}</xsl:text>
327       </xsl:when>
328     </xsl:choose>
329     <xsl:call-template name="colprefix">
330       <xsl:with-param name="loc" select="$loc"/>
331     </xsl:call-template>
332     <xsl:call-template name="colsettings">
333       <xsl:with-param name="loc" select="$loc"/>
334     </xsl:call-template>
335     <xsl:call-template name="colsuffix">
336       <xsl:with-param name="loc" select="$loc"/>
337     </xsl:call-template>
338     <xsl:value-of select="$shield"/>
339     <xsl:text>&#xa;</xsl:text>
340     <xsl:value-of select="$shield"/>
341     <xsl:text>   </xsl:text>
342   </xsl:for-each>
343   <xsl:choose>
344     <xsl:when test="@tgroupstyle='overhang'">
345       <xsl:text></xsl:text>
346     </xsl:when>
347     <xsl:otherwise>
348       <xsl:text>@{</xsl:text>
349     </xsl:otherwise>
350   </xsl:choose>
351   <xsl:text>}&#xa;</xsl:text>

```

```

352 </xsl:template>
353
354 <!-- don't process the colspecs inline: they get managed as above -->
355 <xsl:template match="db:colspec"/>

```

loc Experimental. The \$loc parameter is tested in the column handler named templates for the value "thead" in order to prevent typographic modification of the column header text.

colprefix This template handles the prefixing of a \LaTeX column specification with default formatting as provided for by the array package:

- @align SHOULD specify the column alignment (left/right/center). The default is left-aligned
- @colwidth MAY specify the column width in some units or as a percentage (using the \% after the number)
- @wordsize MAY specify the font size as two numbers separated by a slash (font size and baseline spacing)
- @condition MAY specify additional \LaTeX formatting commands

```

356 <xsl:template name="colprefix">
357   <xsl:param name="loc">
358     <xsl:text></xsl:text>
359   </xsl:param>
360   <!-- @colwidth in <colspec>
361        @condition for fonts, uses LaTeX code
362        @wordsize in <entry>, here and below
363        BUT header cells with no attributes
364        masquerade as their colspec to get the
365        settings, but set $loc to 'thead' -->
366   <xsl:if test="@arch='leftrule'">
367     <xsl:text>|</xsl:text>
368   </xsl:if>
369   <xsl:if test="@condition or @colwidth or @wordsize
370              or ancestor::db:thead or $loc='thead'">
371     <xsl:text>>{</xsl:text>
372     <xsl:variable name="nth">
373       <xsl:choose>
374         <xsl:when test="name()='colspec'">
375           <xsl:value-of
376             select="count(preceding-sibling::db:colspec)+1"/>
377         </xsl:when>
378         <xsl:otherwise>
379           <xsl:value-of
380             select="count(preceding-sibling::db:entry)+1"/>
381         </xsl:otherwise>
382       </xsl:choose>
383     </xsl:variable>
384     <xsl:choose>
385       <!-- go raggedright by default and
386            narrow the linespacing for multiline columns -->
387     <xsl:when
388       test="(@align='left'

```

```

389         or
390         ancestor::db:tgroup/db:colspec
391         [position()=$nth]/@align='left')
392     and
393     (@colwidth
394     or
395     @wordsize
396     or
397     ancestor::db:thead or $loc='thead')">
398     <xsl:text>\renewcommand{\baselinestretch}{0.8}</xsl:text>
399     <xsl:text>\selectfont{}\raggedright{}\</xsl:text>
400 </xsl:when>
401 <xsl:when
402     test="(@align='right'
403     or
404     ancestor::db:tgroup/db:colspec
405     [position()=$nth]/@align='right')
406     and
407     (@colwidth
408     or
409     @wordsize
410     or
411     ancestor::db:thead or $loc='thead')">
412     <xsl:text>\renewcommand{\baselinestretch}{0.8}</xsl:text>
413     <xsl:text>\selectfont{}\raggedleft{}\</xsl:text>
414 </xsl:when>
415 <xsl:when test="(@align='center'
416     or
417     ancestor::db:tgroup/db:colspec
418     [position()=$nth]/@align='center')
419     and
420     (@colwidth
421     or
422     @wordsize
423     or
424     ancestor::db:thead or $loc='thead')">
425     <xsl:text>\renewcommand{\baselinestretch}{0.8}</xsl:text>
426     <xsl:text>\selectfont{}\centering{}\</xsl:text>
427 </xsl:when>
428 <!-- default is justified anyway,
429     and char is inapplicable -->
430 <xsl:when test="ancestor::db:thead">
431     <xsl:text>\renewcommand{\baselinestretch}{0.8}</xsl:text>
432     <xsl:text>\selectfont{}\raggedright{}\</xsl:text>
433 </xsl:when>
434 </xsl:choose>
435 <!-- condition is for typographic changes -->
436 <xsl:if test="not($loc='thead')">
437     <xsl:value-of select="@condition"/>
438     <xsl:if test="@colwidth">
439         <xsl:text>\renewcommand{\baselinestretch}{0.8}</xsl:text>
440         <xsl:text>\selectfont{}\</xsl:text>
441     </xsl:if>
442 </xsl:if>

```

```

443     <xsl:text>\CPKprestrut</xsl:text>
444     <xsl:if
445       test="@wordsize and not($loc='thead')">
446       <xsl:text>\fontsize{</xsl:text>
447       <xsl:value-of
448         select="tokenize(@wordsize,'')[1]">
449       <xsl:text>}{</xsl:text>
450       <xsl:value-of
451         select="tokenize(@wordsize,'')[2]">
452       <xsl:text>}\selectfont</xsl:text>
453     </xsl:if>
454     <xsl:if
455       test="@align
456         and
457         (
458           (name()='colspec'
459             and
460             count(following-sibling::db:colspec)=0)
461         or
462         (name()='entry'
463           and
464           count(following-sibling::db:entry)=0)
465         )
466         and
467         (@colwidth or @wordsize or @condition)">
468     <xsl:text>\arraybackslash</xsl:text>
469     <!--
470     <xsl:text>align=</xsl:text>
471     <xsl:value-of select="@align"/>
472     <xsl:text>,name=</xsl:text>
473     <xsl:value-of select="name()"/>
474     <xsl:text>,cols=</xsl:text>
475     <xsl:value-of select="count(following-sibling::db:colspec)"/>
476     <xsl:text>,entries=</xsl:text>
477     <xsl:value-of select="count(following-sibling::db:entry)"/>
478     <xsl:text>,width=</xsl:text>
479     <xsl:value-of select="@colwidth"/>
480     <xsl:text>,size=</xsl:text>
481     <xsl:value-of select="@wordsize"/>
482     <xsl:text>,cond=</xsl:text>
483     <xsl:value-of select="@condition"/>
484     -->
485     </xsl:if>
486     <xsl:text>}</xsl:text>
487   </xsl:if>
488 </xsl:template>

```

The final (only) column specification (or heading \multicolumn) gets the command \arraybackslash to prevent formatting leaking into the column management environment.

colsettings This template sets the column type (left/right/centered) and vertical alignment if the width is specified. Headings are always aligned on the baseline, to allow

multi-word headings to stack properly; otherwise the value of the @char attribute is used: "m" or "b" for middle-alignment or bottom-alignment; the default is alignment on the top line of the cell.

The @colwidth value (in length units or a percent) specifies a paragraphic (potentially multi-line) column; a value of "widest" specifies a setting of the widest value in the row entries, taken as the content passed to the \widthof command.

```

489 <xsl:template name="colsettings">
490   <xsl:param name="loc">
491     <xsl:text></xsl:text>
492   </xsl:param>
493   <!-- this is the type of column (p|mb) and width -->
494   <xsl:choose>
495     <xsl:when test="@colwidth">
496       <xsl:choose>
497         <!-- for db:thead it's always 'b' -->
498         <xsl:when test="$loc='thead'">
499           <xsl:text>b</xsl:text>
500         </xsl:when>
501         <xsl:when test="@char='m' or @char='b'">
502           <xsl:value-of select="@char"/>
503         </xsl:when>
504         <xsl:otherwise>
505           <xsl:text>p</xsl:text>
506         </xsl:otherwise>
507       </xsl:choose>
508       <xsl:text>{</xsl:text>
509       <xsl:choose>
510         <xsl:when test="@colwidth='widest'">
511           <xsl:text>\widthof{</xsl:text>
512           <xsl:call-template name="getlongest"/>
513           <xsl:text>}</xsl:text>
514         </xsl:when>
515         <xsl:when test="contains(@colwidth,'%')">
516           <xsl:value-of
517             select="number(substring-before(
518               @colwidth,'%'))
519             div 100"/>
520           <xsl:text>\columnwidth</xsl:text>
521         </xsl:when>
522         <xsl:otherwise>
523           <xsl:value-of select="@colwidth"/>
524         </xsl:otherwise>
525       </xsl:choose>
526       <xsl:text>}</xsl:text>
527     </xsl:when>
528     <!-- otherwise width not specified -->
529     <xsl:otherwise>
530       <xsl:choose>
531         <!-- for db:thead it's always 'b' -->
532         <xsl:when test="$loc='thead'">
533           <xsl:text>b{</xsl:text>
534           <xsl:call-template name="getlongest"/>

```

```

535         <!-- make this the longest tbody entry, not thead -->
536
537         <!-- this doesn't work:
538         <xsl:value-of
539             select="following-sibling::db:tbody
540                     /db:row/db:entry[position()=$nth]
541                     [not(..//db:row/db:entry[position()=$nth]
542                         /string-length()
543                         > current()/string-length())]" />
544         -->
545         <!-- and Saxon PE needed for this one:
546         select="sort(following-sibling::db:tbody
547                     /db:row/db:entry[position()=$nth],
548                     function($c)
549                         {string-length($c)})[last()]" />
550         -->
551
552         <!-- append 's' to account for
553             extra width of bold type 2020-04-24 -->
554         <xsl:text>s}}</xsl:text>
555     </xsl:when>
556     <xsl:when test="@align='left' or
557                 @align='right' or
558                 @align='center'">
559         <xsl:value-of
560             select="substring(@align,1,1)" />
561     </xsl:when>
562     <xsl:when test="@align='char'">
563         <xsl:text>d</xsl:text>
564     </xsl:when>
565     <!-- justified is implied by
566         @colwidth above -->
567 </xsl:choose>
568 </xsl:otherwise>
569 </xsl:choose>
570 </xsl:template>

```

Experimental: an @align value of "char" specifies decimal alignment.

`colsuffix` This template implements the value of the @conformance attribute (usually a \LaTeX command) on a <colspec> or header <entry> element as a suffix to the column specification.

```

571 <xsl:template name="colsuffix">
572     <xsl:param name="loc">
573         <xsl:text></xsl:text>
574     </xsl:param>
575     <xsl:if test="@conformance
576                 or
577                 (@align!='left'
578                 and
579                 @align!='right'
580                 and
581                 @align!='center')

```

```

582             or
583             count(following-sibling::db:colspec)=0">
584     <xsl:text>&lt;{</xsl:text>
585     <xsl:value-of
586       select="normalize-space(@conformance)"/>
587     <xsl:if test="@align!='left' and
588               @align!='right' and
589               @align!='center'">
590       <xsl:text>\CPKpoststrut</xsl:text>
591     </xsl:if>
592     <!-- why was this needed in colsuffixes?
593     <xsl:if test="count(following-sibling::db:colspec)=0">
594       <xsl:text>\arraybackslash</xsl:text>
595     </xsl:if>
596     -->
597     <xsl:text>}</xsl:text>
598   </xsl:if>
599 </xsl:template>

```

thead The table header (column headers row) is terminated by 2pt extra space and a horizontal rule across all columns.

```

600 <xsl:template match="db:thead">
601   <xsl:apply-templates/>
602   <xsl:text>[2pt]\hrline</xsl:text>
603   <xsl:if test="ancestor::db:table/@condition='long'">
604     <xsl:text>\endhead</xsl:text>
605   </xsl:if>
606   <xsl:text>&#xa;</xsl:text>
607 </xsl:template>

```

tbody The table body contains just rows. The first one gets a \vstrut for spacing away from the rule just issued by the <thead> unless the cell is the start of a multi-row span. The specialist code for embedding data from the "data" part is experimental and not for normal use.

```

608 <xsl:template match="db:tbody">
609   <xsl:value-of select="$shield"/>
610   <xsl:text> </xsl:text>
611   <xsl:if test="not(db:row[1]/db:entry[1]/@spanname)">
612     <xsl:text>\CPKvstrut </xsl:text>
613   </xsl:if>
614   <xsl:if test="ancestor::db:table/@condition='long'
615             and
616             ancestor::db:table/@xml:id">
617     <xsl:text>\label{</xsl:text>
618     <xsl:value-of select="ancestor::db:table/@xml:id"/>
619     <xsl:text>}%</xsl:text>
620   </xsl:if>
621   <xsl:text>&#xa;</xsl:text>
622   <!-- obsolete, was used in ucc thesis
623       eg, xlink:href='deg-acsss'
624       to use the external tables

```

```

625     <xsl:choose>
626       <xsl:when
627         test="@xlink:href and @xlink:show='embed'">
628         <xsl:apply-templates mode="row"
629           select="//db:part[@xml:id='data']
630             //db:constraintdef[@xml:id=
631               concat(substring-before(
632                 current()/@xlink:href,'-'),
633                 'options')]
634             /db:methodsynopsis
635             [tokenize(@arch,' ')=
636               substring-after(
637                 current()/@xlink:href,'-')]"/>
638         <xsl:sort select="@xml:id"/>
639       </xsl:apply-templates>
640     </xsl:when>
641     <xsl:otherwise>
642       <xsl:apply-templates/>
643     </xsl:otherwise>
644   </xsl:choose>
645   -->
646   <xsl:apply-templates/>
647 </xsl:template>

```

methodsynopsis Experimental. This element holds data for specifying large numbers of academic options for thesis templates.

```

648 <xsl:template match="db:methodsynopsis" mode="row">
649   <!-- each of these produces one row of the table -->
650   <xsl:value-of select="$shield"/>
651   <xsl:text> </xsl:text>
652   <!-- signal a footnote if needed -->
653   <xsl:if test="db:modifier">
654     <xsl:text>\fnote{}</xsl:text>
655   </xsl:if>
656   <!-- col 1: token for option -->
657   <xsl:value-of select="@xml:id"/>
658   <xsl:text>&#xa;</xsl:text>
659   <!-- col 2: division[s],
660     may be multiple methodparam elements -->
661   <xsl:for-each select="db:methodparam/db:parameter">
662     <xsl:value-of select="$shield"/>
663     <xsl:text>&#x9;</xsl:text>
664     <!-- @remap is a prefix
665       except for institutes and centres -->
666     <xsl:if test="not(@remap='') and
667       @role!='institute' and
668       @role!='centre'">
669       <xsl:value-of
670         select="normalize-space(@remap)"/>
671       <xsl:text> </xsl:text>
672     </xsl:if>
673     <!-- the name -->
674     <xsl:apply-templates/>

```

```

675      <!-- @remap is a suffix for
676           institutes and centres -->
677      <xsl:if test="not(@remap='') and
678                  (@role='institute' or
679                  @role='centre')">
680          <xsl:text> </xsl:text>
681          <xsl:value-of
682              select="normalize-space(@remap)"/>
683      </xsl:if>
684      <!-- any following divisions involved
685           (only for affiliations) -->
686      <xsl:choose>
687          <xsl:when test="position()=last()">
688              <xsl:text>; </xsl:text>
689          </xsl:when>
690          <!-- dummy to terminate
691               degree options cleanly -->
692          <xsl:when test="@role='degree'">
693              <xsl:text></xsl:text>
694          </xsl:when>
695          <xsl:otherwise>
696              <xsl:text>&#x9;</xsl:text>
697          </xsl:otherwise>
698      </xsl:choose>
699  </xsl:for-each>
700  <!-- citation format -->
701  <xsl:if
702      test="not(db:methodparam/db:parameter
703                /@role='degree')">
704      <xsl:value-of select="$shield"/>
705      <xsl:text>&#x9;</xsl:text>
706      <xsl:value-of select="db:methodname"/>
707  </xsl:if>
708  <xsl:text>\\hline[.1pt]&#x9;</xsl:text>
709  </xsl:template>

```

row Table rows are output to terminate with the \LaTeX double backslash as normal. Attributes in the `<row>` element control usage:

- `@role="header"` creates 2pt more space above and below the row
- `@rowsep="1"` add 1ex of additional space below the row
- `@remap="newpage"` causes the table to be stopped and restarted with repeated headings over a page-break
- `@condition="test"`, when the sole content of the first `<entry>` cell is a marked-up element type name, causes the row to be included when the named element occurs at least once in the main document
- If the `<tbody>` attribute `@style` is set to "ruled" the table will be ruled horizontally and vertically

```

710  <!-- omit rows with @condition and some weird zero value -->
711  <xsl:template

```

```

712     match="db:row[@condition
713         and
714         count($thisdoc//*[
715             [name()='name(current()/db:entry[1]/*[1])]
716             = 0]"/>
717 <!-- normal rows -->
718 <xsl:template
719     match="db:row[not(@condition)
720         or
721         count($thisdoc//*[
722             [name()='name(current()/db:entry[1]/*[1])]
723             > 0]"/>
724 <xsl:value-of select="$shield"/>
725 <xsl:text> </xsl:text>
726 <xsl:if test="@role='header'">
727     <!-- extra space except on first row -->
728     <xsl:if test="count(preceding-sibling::db:row) > 1">
729         <xsl:text>[2pt]\hline&#xa;</xsl:text>
730         <xsl:value-of select="$shield"/>
731         <xsl:text> </xsl:text>
732     </xsl:if>
733 </xsl:if>
734 <xsl:if test="@role='groupstart'">
735     <xsl:if test="count(preceding-sibling::db:row) > 1">
736         <xsl:text>[2pt]\hline\hline&#xa;</xsl:text>
737         <xsl:value-of select="$shield"/>
738         <xsl:text> </xsl:text>
739     </xsl:if>
740 </xsl:if>
741 <xsl:apply-templates/>
742 <xsl:text>\\</xsl:text>
743 <xsl:choose>
744     <xsl:when test="starts-with(@role,'sep')">
745         <xsl:text>[</xsl:text>
746         <xsl:value-of select="substring-after(@role,'sep')"/>
747         <xsl:text>]</xsl:text>
748     </xsl:when>
749     <xsl:when test="@role='header'">
750         <xsl:text>[2pt]\hline\CPKvstrut</xsl:text>
751     </xsl:when>
752     <xsl:when
753         test="(ancestor::db:tbody or ancestor::db:tfoot)
754             and position()=last()">
755         <xsl:text>[2pt]\hline</xsl:text>
756     </xsl:when>
757     <xsl:when
758         test="parent::*[1]/db:row[1][@rowsep='1']">
759         <xsl:text>[1ex]</xsl:text>
760     </xsl:when>
761     <xsl:when
762         test="ancestor::db:tbody/@style='ruled'">
763         <xsl:text>\hline</xsl:text>
764     </xsl:when>
765 </xsl:choose>

```

```

766     <xsl:if test="not(ancestor::db:thead)">
767       <xsl:text>&#xa;</xsl:text>
768     </xsl:if>
769     <xsl:if test="@remap='newpage'">
770       <xsl:value-of select="$shield"/>
771       <xsl:text> \end{tabular}\par\vfill\flushright</xsl:text>
772       <xsl:text>[continued...\par\endgroup\clearpage&#xa;</xsl:text>
773       <xsl:for-each select="ancestor::db:tgroup[1]">
774         <xsl:call-template name="tabstart"/>
775       </xsl:for-each>
776       <xsl:for-each select="ancestor::db:tgroup/db:thead[1]">
777         <xsl:apply-templates/>
778         <xsl:text>[2pt]\hline</xsl:text>
779         <xsl:if test="ancestor::db:table/@condition='long'">
780           <xsl:text>\endhead</xsl:text>
781         </xsl:if>
782         <xsl:text>&#xa;</xsl:text>
783       </xsl:for-each>
784     </xsl:if>
785   </xsl:template>

```

entryheadfoot Table cells (the `<entry>` elements) in the `<thead>` and `<tfoot>` are turned into `\multicolumns` because they may contain heading text longer than the normal content of a body cell, and therefore need to wrap to extra lines. This uses the same code as for the standard table startup, using bottom alignment.

```

786   <xsl:template
787     match="db:entry[ancestor::db:thead or
788       ancestor::db:tfoot]">
789     <xsl:if test="preceding-sibling::db:entry">
790       <xsl:text>&amp;&#xa;</xsl:text>
791       <xsl:value-of select="$shield"/>
792       <xsl:text> </xsl:text>
793     </xsl:if>
794     <!-- allow empty headers -->
795     <xsl:if test="normalize-space(.)!=''">
796       <!-- identify the column number -->
797       <xsl:variable name="colpos"
798         select="count(preceding-sibling::db:entry) + 1"/>
799       <!-- ALWAYS make headers paragraphic,
800         which means \multicolumn{1}...
801         because we don't know how
802         they need to be formatted -->
803       <xsl:text>\multicolumn{</xsl:text>
804       <xsl:choose>
805         <xsl:when test="@spanname">
806           <xsl:value-of select="@spanname"/>
807         </xsl:when>
808         <xsl:otherwise>
809           <xsl:text>1</xsl:text>
810         </xsl:otherwise>
811       </xsl:choose>
812       <xsl:text>}}</xsl:text>

```

```

813 <!-- omit LH shoulder on first column -->
814 <xsl:if test="$colpos=1">
815   <xsl:text>@{}</xsl:text>
816 </xsl:if>
817 <xsl:choose>
818   <!-- if alignment or width is specified
819         in the heading cell itself,
820         use local settings,
821         honouring local prefix/suffix -->
822   <xsl:when test="@align or @colwidth or @wordsize">
823     <xsl:call-template name="colprefix">
824       <xsl:with-param name="loc" select="''thead''"/>
825     </xsl:call-template>
826     <xsl:call-template name="colsettings">
827       <xsl:with-param name="loc" select="''thead''"/>
828     </xsl:call-template>
829     <xsl:call-template name="colsuffix">
830       <xsl:with-param name="loc" select="''thead''"/>
831     </xsl:call-template>
832   </xsl:when>
833   <!-- otherwise use the column default,
834         but use local prefix/suffix by preference -->
835   <xsl:otherwise>
836     <!-- test for the prefix -->
837     <xsl:choose>
838       <xsl:when
839         test="ancestor::db:tgroup/db:colspec
840              [$colpos]/@condition
841              or
842              ancestor::db:tgroup/db:colspec
843              [$colpos]/@colwidth
844              or
845              ancestor::db:tgroup/db:colspec
846              [$colpos]/@wordsize">
847         <xsl:for-each
848           select="ancestor::db:tgroup/db:colspec
849                 [$colpos]">
850           <xsl:call-template name="colprefix">
851             <xsl:with-param name="loc"
852               select="''thead''"/>
853           </xsl:call-template>
854         </xsl:for-each>
855       </xsl:when>
856       <xsl:otherwise>
857         <xsl:call-template name="colprefix"/>
858       </xsl:otherwise>
859     </xsl:choose>
860     <!-- switch context to the colspec
861           to get the settings -->
862     <xsl:for-each
863       select="ancestor::db:tgroup/db:colspec
864             [$colpos]">
865       <xsl:call-template name="colsettings">
866         <xsl:with-param name="loc"

```

```

867         select="''thead''"/>
868     </xsl:call-template>
869 </xsl:for-each>
870 <!-- test for the suffix -->
871 <xsl:choose>
872     <xsl:when
873         test="ancestor::db:tgroup/db:colspec
874             [position()=$colpos]/@condition
875             or
876             ancestor::db:tgroup/db:colspec
877             [position()=$colpos]/@colwidth
878             or
879             ancestor::db:tgroup/db:colspec
880             [position()=$colpos]/@wordsize">
881         <xsl:for-each
882             select="ancestor::db:tgroup/db:colspec
883                 [position()=$colpos]">
884             <xsl:call-template name="colsuffix"/>
885         </xsl:for-each>
886     </xsl:when>
887     <xsl:otherwise>
888         <xsl:call-template name="colsuffix"/>
889     </xsl:otherwise>
890 </xsl:choose>
891 </xsl:otherwise>
892 </xsl:choose>
893 <xsl:if
894     test="count(following-sibling::db:entry)=0">
895     <xsl:text>@{</xsl:text>
896 </xsl:if>
897 <xsl:text>}}\sffamily\bfseries </xsl:text>
898 <xsl:apply-templates/>
899 <xsl:text>}}</xsl:text>
900 </xsl:if>
901 </xsl:template>

```

TODO: Entry head/foot code needs moving to a named template

entrybody Cells in the table body are check for multicolumn spans, overhangs, bold for crossheads, and row spans.

```

902 <xsl:template match="db:entry[ancestor::db:tbody]">
903     <xsl:variable name="colpos"
904         select="count(preceding-sibling::db:entry) + 1"/>
905     <xsl:if test="preceding-sibling::db:entry">
906         <xsl:text>&lt;/xsl:text>
907     </xsl:if>
908     <xsl:if test="@spanname">
909         <xsl:text>\multicolumn{</xsl:text>
910         <xsl:value-of select="@spanname"/>
911         <xsl:text>}</xsl:text>
912     </xsl:if>

```

```

913         test="count(preceding-sibling::db:entry)=0">
914         <xsl:text>@{</xsl:text>
915     </xsl:if>
916     <xsl:choose>
917         <!-- if alignment or width is specified,
918             use local settings -->
919         <xsl:when test="@align or @wordsize">
920             <xsl:call-template name="colprefix"/>
921             <xsl:call-template name="colsettings"/>
922             <xsl:call-template name="colsuffix"/>
923         </xsl:when>
924         <!-- otherwise use default,
925             but check local pre and post -->
926         <xsl:otherwise>
927             <xsl:call-template name="colprefix"/>
928             <xsl:for-each
929                 select="ancestor::db:tgroup/db:colspec
930                     [position()=$colpos]">
931                 <xsl:call-template name="colsettings"/>
932             </xsl:for-each>
933             <xsl:call-template name="colsuffix"/>
934         </xsl:otherwise>
935     </xsl:choose>
936     <xsl:if
937         test="count(following-sibling::db:entry)=0">
938         <xsl:text>@{</xsl:text>
939     </xsl:if>
940     <xsl:text>}}</xsl:text>
941 </xsl:if>
942 <!-- check for hangouts -->
943 <xsl:if test="@annotations">
944     <xsl:text>\llap{\normalfont </xsl:text>
945     <xsl:value-of select="@annotations"/>
946     <xsl:text>}</xsl:text>
947 </xsl:if>
948 <!-- text for headers is in bold -->
949 <xsl:if test="../@role='header'">
950     <xsl:text>\CPKvstrut\bfseries </xsl:text>
951 </xsl:if>
952 <!-- check for row-spanned brace-collected -->
953 <xsl:if test="@morerows">
954     <xsl:if test="@charoff and @char">
955         <xsl:text>\llap{</xsl:text>
956         <xsl:value-of select="@char"/>
957         <xsl:text>{</xsl:text>
958         <xsl:value-of select="@morerows"/>
959         <xsl:text>}}</xsl:text>
960         <xsl:value-of select="@charoff"/>
961         <xsl:text>}</xsl:text>
962         <xsl:text>}}&#xa;</xsl:text>
963         <xsl:value-of select="$shield"/>
964         <xsl:text> &#x9;</xsl:text>
965     </xsl:if>
966     <xsl:text>\multirow{</xsl:text>

```

```

967     <xsl:value-of select="@morerows"/>
968     <xsl:text>}{</xsl:text>
969     <xsl:value-of select="@wordsize"/>
970     <xsl:text>}{</xsl:text>
971 </xsl:if>
972 <!-- now we're in the content at last -->
973 <xsl:if test="@condition">
974     <xsl:value-of select="@condition"/>
975     <xsl:text>{</xsl:text>
976 </xsl:if>
977 <xsl:choose>
978     <!-- mldr &#x2026; on its own (\dotfill)
979         needs a \hbox -->
980     <xsl:when test=".\dotfill{}' and @wordsize">
981         <xsl:text>\hbox to </xsl:text>
982         <xsl:value-of select="@wordsize"/>
983         <xsl:text>{</xsl:text>
984         <xsl:value-of select="."/>
985         <xsl:text>}</xsl:text>
986     </xsl:when>
987     <!-- hardwired exceptions to special characters -->
988     <xsl:when test="@conformance">
989         <xsl:value-of select="@conformance"/>
990     </xsl:when>
991     <xsl:otherwise>
992         <xsl:apply-templates/>
993     </xsl:otherwise>
994 </xsl:choose>
995 <!-- additional manual extras before end of cell -->
996 <xsl:value-of select="@userlevel"/>
997 <!-- end of entry, before any grouping -->
998 <xsl:if
999     test="ancestor::db:tbody/db:row[1][@rowsep='1']
1000         and
1001         ancestor::db:tgroup/db:colspec
1002         [position()=$colpos][@colwidth]
1003         and
1004         not(position()=last())">
1005     <xsl:text>\vrule depth1em width0pt</xsl:text>
1006 </xsl:if>
1007 <xsl:if test="@morerows">
1008     <xsl:text>}</xsl:text>
1009 </xsl:if>
1010 <xsl:if test="@spanname">
1011     <xsl:text>}</xsl:text>
1012 </xsl:if>
1013 </xsl:template>

```

Figures

figure A whole-page PDF illustration gets a forced page-break before and after, to ensure it occurs where it is requested, rather than floating to later. The remainder of the figure environment is constructed similarly to the tables, but any changes

to fonts are left to the content.

```

1014 <xsl:template match="db:figure">
1015   <xsl:if
1016     test="@floatstyle='p'
1017       and
1018       descendant::db:imagedata
1019         [@format='pdf' and @condition]">
1020     <xsl:value-of select="$shield"/>
1021     <xsl:text> \clearpage&#xa;</xsl:text>
1022   </xsl:if>
1023   <xsl:value-of select="$shield"/>
1024   <xsl:text> \begin{figure}[</xsl:text>
1025   <xsl:choose>
1026     <xsl:when test="@floatstyle">
1027       <xsl:value-of select="@floatstyle"/>
1028     </xsl:when>
1029     <xsl:otherwise>
1030       <xsl:text>ht</xsl:text>
1031     </xsl:otherwise>
1032   </xsl:choose>
1033   <xsl:text>]</xsl:text>
1034   <xsl:text>\small\sffamily&#xa;</xsl:text>
1035   <xsl:apply-templates/>
1036   <xsl:value-of select="$shield"/>
1037   <xsl:text> \end{figure}&#xa;</xsl:text>
1038   <xsl:if test="@floatstyle='p'
1039     and
1040     descendant::db:imagedata
1041       [@format='pdf' and @condition]">
1042     <xsl:value-of select="$shield"/>
1043     <xsl:text> \clearpage&#xa;</xsl:text>
1044   </xsl:if>
1045 </xsl:template>

```

`informalfigure` An informal figure is just a centred image.

```

1046 <xsl:template match="db:informalfigure">
1047   <xsl:value-of select="$shield"/>
1048   <xsl:text> \begin{center}&#xa;</xsl:text>
1049   <xsl:apply-templates/>
1050   <xsl:value-of select="$shield"/>
1051   <xsl:text> \end{center}&#xa;</xsl:text>
1052 </xsl:template>

```

`mediaobject` Images themselves can be framed, and the width adjusted for the border and rule thickness.

```

1053 <xsl:template match="db:mediaobject
1054   [not(parent::db:figure)]">
1055   <xsl:apply-templates/>
1056 </xsl:template>
1057
1058 <xsl:template match="db:figure/db:mediaobject |

```

```

1059             db:informalfigure/db:mediaobject">
1060     <xsl:if test="@role='framed'">
1061       <xsl:value-of select="$shield"/>
1062       <xsl:text> \fbox{\vbox{%</xsl:text>
1063       <xsl:value-of select="$shield"/>
1064       <xsl:text> \advance\hsize by-2\fboxsep</xsl:text>
1065       <xsl:text>\advance\hsize by-2\fboxrule&#xa;</xsl:text>
1066     </xsl:if>
1067     <xsl:apply-templates/>
1068     <xsl:if test="@role='framed'">
1069       <xsl:value-of select="$shield"/>
1070       <xsl:text> }}&#xa;</xsl:text>
1071     </xsl:if>
1072   </xsl:template>

```

`imageobject` This element controls for the presence of multiple images, which can be stacked left-to-right, or allowed to occur vertically.

```

1073   <xsl:template match="db:imageobject">
1074     <xsl:choose>
1075       <!-- detect a previous image -->
1076       <xsl:when test="preceding-sibling::*[1]
1077         [local-name()='imageobject']">
1078         <xsl:choose>
1079           <xsl:when test="@dir='ltr'">
1080             <xsl:value-of select="$shield"/>
1081             <xsl:text> \quad\vrule\quad&#xa;</xsl:text>
1082           </xsl:when>
1083           <xsl:when test="ancestor::db:figure">
1084             <xsl:value-of select="$shield"/>
1085             <xsl:text> \par&#xa;</xsl:text>
1086           </xsl:when>
1087           <xsl:otherwise>
1088             <xsl:value-of select="$shield"/>
1089             <xsl:text> \&#xa;</xsl:text>
1090           </xsl:otherwise>
1091         </xsl:choose>
1092       </xsl:when>
1093       <xsl:otherwise>
1094         <xsl:text></xsl:text>
1095       </xsl:otherwise>
1096     </xsl:choose>
1097     <xsl:apply-templates/>
1098   </xsl:template>

```

`imagedata` The `<imagedata>` specifies the image file, alignment, framing of non-PDF images, width, depth, and scale.

The default alignment, like tables, is centering. Images may be framed by setting the `@arch` to "framed".

```

1099   <xsl:template match="db:imagedata">
1100     <!-- alignment -->

```

```

1101     <xsl:choose>
1102       <xsl:when test="@align='center' or not(@align)">
1103         <xsl:value-of select="$shield"/>
1104         <xsl:text> \centering&#xa;</xsl:text>
1105       </xsl:when>
1106       <xsl:when test="@align='right'">
1107         <xsl:value-of select="$shield"/>
1108         <xsl:text> \flushright&#xa;</xsl:text>
1109       </xsl:when>
1110     </xsl:choose>
1111     <!-- reframe non-PDFs ??? -->
1112     <xsl:if test="@arch='framed' and not(@format='pdf')">
1113       <xsl:value-of select="$shield"/>
1114       <xsl:text> </xsl:text>
1115       <xsl:if test="@contentdepth">
1116         <xsl:text>\fboxsep</xsl:text>
1117         <xsl:value-of select="@contentdepth"/>
1118       </xsl:if>
1119       <xsl:text>\fbox{&#xa;</xsl:text>
1120     </xsl:if>

```

imagedatafiles The @fileref attribute holds the filename. PDF page images are included with the \includepdf command from the pdfpages package; otherwise the normal \includegraphics command from the graphicx package is used. The @condition can be used to hold optional arguments for the \includepdf command for PDF images; otherwise the @width and @depth attributes are used for width and height (in units or percent of the textwidth or textheight).

```

1121     <xsl:choose>
1122       <xsl:when test="@fileref">
1123         <xsl:choose>
1124           <xsl:when test="@format='pdf' and @condition">
1125             <xsl:value-of select="$shield"/>
1126             <xsl:text> \includepdf</xsl:text>
1127           </xsl:when>
1128           <xsl:otherwise>
1129             <xsl:value-of select="$shield"/>
1130             <xsl:text> \includegraphics</xsl:text>
1131           </xsl:otherwise>
1132         </xsl:choose>
1133       <xsl:text>[</xsl:text>
1134       <xsl:choose>
1135         <xsl:when test="@condition">
1136           <xsl:value-of
1137             select="normalize-space(@condition)"/>
1138         <xsl:if
1139           test="@arch='framed' and @format='pdf'">
1140           <xsl:text>,frame</xsl:text>
1141         </xsl:if>
1142       </xsl:when>
1143       <xsl:when test="@width">
1144         <xsl:text>width=</xsl:text>
1145       <xsl:choose>

```

```

1146         <xsl:when test="contains(@width,'%')">
1147             <xsl:value-of
1148                 select="number(substring-before
1149                     (@width,'%'))
1150                     div 100"/>
1151             <xsl:text>\columnwidth</xsl:text>
1152         </xsl:when>
1153         <xsl:otherwise>
1154             <xsl:value-of select="@width"/>
1155         </xsl:otherwise>
1156     </xsl:choose>
1157 </xsl:when>
1158 <xsl:when test="@depth">
1159     <xsl:text>height=</xsl:text>
1160     <xsl:choose>
1161         <xsl:when test="contains(@depth,'%')">
1162             <xsl:value-of
1163                 select="number(substring-before
1164                     (@depth,'%'))
1165                     div 100"/>
1166             <xsl:text>\textheight</xsl:text>
1167         </xsl:when>
1168         <xsl:otherwise>
1169             <xsl:value-of select="@depth"/>
1170         </xsl:otherwise>
1171     </xsl:choose>
1172 </xsl:when>
1173 <xsl:when test="@scalefit='1'">
1174     <xsl:text>width=.5\columnwidth</xsl:text>
1175 </xsl:when>
1176 <xsl:otherwise>
1177     <xsl:variable name="numpix"
1178         select="count(ancestor::db:mediaobject
1179             /db:imageobject)"/>
1180     <xsl:variable name="numpixfrac"
1181         select="1 div $numpix"/>
1182     <xsl:text>width=</xsl:text>
1183     <xsl:value-of select="$numpixfrac"/>
1184     <xsl:text>\columnwidth</xsl:text>
1185     <!-- allow for the \quad\vrule\quad
1186         between multiple images
1187         for which we use the
1188         notional value 2.2em -->
1189     <xsl:if test="$numpix>1">
1190         <xsl:text> - </xsl:text>
1191         <xsl:value-of select="($numpix - 1)*2.2"/>
1192         <xsl:text>em</xsl:text>
1193     </xsl:if>
1194     <xsl:if test="@arch='framed'">
1195         <xsl:text> - 2\fboxsep - 1\fboxrule</xsl:text>
1196     </xsl:if>
1197 </xsl:otherwise>
1198 </xsl:choose>
1199 <xsl:text>]</xsl:text>

```

```

1200      <xsl:text>{</xsl:text>
1201      <xsl:value-of select="@fileref"/>
1202      <xsl:text>}</xsl:text>
1203      </xsl:when>

```

In the case of multiple `<imageobject>` elements for multiple images, they are scaled to fit side-by-side.

`imagedataremap` If no filename is specified, the contents of the `@condition` and `@remap` attributes are output, if present; otherwise the word 'IMAGE' is used as a placeholder.

```

1204      <xsl:when test="@remap">
1205        <xsl:value-of select="$shield"/>
1206        <xsl:text> </xsl:text>
1207        <xsl:if test="@condition">
1208          <xsl:text>{</xsl:text>
1209          <xsl:value-of
1210            select="normalize-space(@condition)"/>
1211          </xsl:if>
1212          <xsl:value-of select="@remap"/>
1213          <xsl:if test="@condition">
1214            <xsl:text>}</xsl:text>
1215          </xsl:if>
1216        </xsl:when>
1217        <xsl:otherwise>
1218          <xsl:text>IMAGE</xsl:text>
1219        </xsl:otherwise>
1220      </xsl:choose>
1221      <xsl:if test="@arch='framed'
1222        and
1223        not(@format='pdf')">
1224        <xsl:text>}</xsl:text>
1225      </xsl:if>
1226      <xsl:text>&#xa;</xsl:text>
1227    </xsl:template>

```

`textobject` Textual figure are done with the `<textobject>` element.

```

1228    <xsl:template match="db:textobject">
1229      <xsl:text>\\[3pt]\raggedright&#xa;</xsl:text>
1230      <xsl:apply-templates/>
1231    </xsl:template>
1232
1233  </xsl:stylesheet>

```

14 The db2md.xsl script

These are templates used to generate the README.md file required by CTAN submissions. They use the boilerplate text in readme.xml. This tries to follow the CTAN requirements documented at <https://ctan.org/help/markdown/style>.

db2mdscript-comment Identify the file origin, date, time, etc.

```
1 <!-- The db2md.xsl script (from the classpack package v.1.28, 2024-02-21) -->
2 <!-- Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16 -->
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

14.1 XML Declaration, namespaces, and setup

The namespaces are the same as for the calling routine, db2dtx.xsl.

init Set up the namespaces, the same as for the main script.

```
3 <xsl:stylesheet
4   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
5   xmlns:db="http://docbook.org/ns/docbook"
6   xmlns:xlink="http://www.w3.org/1999/xlink"
7   xmlns:sil="http://xml.silmaril.ie/identity/functions"
8   xmlns:xs="http://www.w3.org/2001/XMLSchema"
9   version="3.0">
```

setup Use the current context to remember what the main document was; and be nice to editors with the line width.

```
10 <xsl:variable name="maindoc" select="."/>
11
12 <xsl:variable name="width">
13   <xsl:text>72</xsl:text>
14 </xsl:variable>
```

plaintext [db:title with [not(parent::db:chapter) and not(parent::db:sect1) and not(parent::db:sect2) and not(parent::db:sect3) and not(parent::db:warning) and not(parent::db:formalpara)] and db:para with [not(parent::db:listitem) and not(ancestor::db:variablelist) and not(parent::db:warning) and not(parent::db:formalpara)] and db:remark]

Plain titles and plain paras and remarks (used for documentation of the documentation). Note that Abstract paragraphs match this template.

```
15 <xsl:template mode="readme"
16   match="db:title[not(parent::db:chapter) and
17   not(parent::db:sect1) and
18   not(parent::db:sect2) and
```

```

19             not(parent::db:sect3) and
20             not(parent::db:warning) and
21             not(parent::db:formalpara)]
22         |
23         db:para[not(parent::db:listitem) and
24             not(parent::db:warning) and
25             not(parent::db:formalpara)]
26         |
27         db:remark">
28     <xsl:variable name="content">
29         <xsl:apply-templates select="node()" mode="inline"/>
30     </xsl:variable>
31     <!-- blank line (previously elided in rendering MD???) -->
32     <xsl:text>&#xa;</xsl:text>
33     <!-- call the prefix-generator to handle the flag -->
34     <xsl:call-template name="markup"/>
35     <xsl:call-template name="normtext">
36         <xsl:with-param name="content"
37             select="normalize-space($content)"/>
38         <xsl:with-param name="indent">
39             <xsl:if
40                 test="ancestor::db:part[@xml:id='code']
41                     or
42                     ancestor::db:procedure
43                     [@xml:id='prepackage']">
44                 <xsl:value-of select="$sentinel"/>
45             </xsl:if>
46         </xsl:with-param>
47     </xsl:call-template>
48     <xsl:if test="ancestor::db:part[@xml:id!='code']">
49         <xsl:text>&#xa;</xsl:text>
50     </xsl:if>
51 </xsl:template>

```

db:title [in db:chapter/ and db:sect1/db:title and db:sect2/db:title and db:sect3/db:title and db:bridgehead]

Section titles all work the same way (increasing numbers of hash marks): chapter, section, subsection, subsubsection and bridgeheads. In each case make the content with markup, make the markup, normalize the output, and indent it as needed.

```

52 <xsl:template match="db:chapter/db:title" mode="readme">
53     <xsl:variable name="content">
54         <xsl:apply-templates select="node()" mode="inline"/>
55     </xsl:variable>
56     <xsl:call-template name="markup">
57         <xsl:with-param name="flag" select="'#'" />
58     </xsl:call-template>
59     <xsl:call-template name="normtext">
60         <xsl:with-param name="content"
61             select="normalize-space($content)"/>
62         <xsl:with-param name="indent">
63             <xsl:if

```

```

64         test="ancestor::db:part[@xml:id='code'] or
65         ancestor::db:procedure[@xml:id='prepackage']">
66         <xsl:value-of select="$sentinel"/>
67     </xsl:if>
68 </xsl:with-param>
69 </xsl:call-template>
70 </xsl:template>
71
72 <xsl:template match="db:sect1/db:title" mode="readme">
73     <xsl:variable name="content">
74         <xsl:apply-templates select="node()" mode="inline"/>
75     </xsl:variable>
76     <xsl:call-template name="markup">
77         <xsl:with-param name="flag" select="'##'"/>
78     </xsl:call-template>
79     <xsl:call-template name="normtext">
80         <xsl:with-param name="content"
81             select="normalize-space($content)"/>
82         <xsl:with-param name="indent">
83             <xsl:if
84                 test="ancestor::db:part[@xml:id='code'] or
85                 ancestor::db:procedure[@xml:id='prepackage']">
86                 <xsl:value-of select="$sentinel"/>
87             </xsl:if>
88         </xsl:with-param>
89     </xsl:call-template>
90 </xsl:template>
91
92 <xsl:template match="db:sect2/db:title" mode="readme">
93     <xsl:variable name="content">
94         <xsl:apply-templates select="node()" mode="inline"/>
95     </xsl:variable>
96     <xsl:call-template name="markup">
97         <xsl:with-param name="flag" select="'###'"/>
98     </xsl:call-template>
99     <xsl:call-template name="normtext">
100         <xsl:with-param name="content"
101             select="normalize-space($content)"/>
102         <xsl:with-param name="indent">
103             <xsl:if
104                 test="ancestor::db:part[@xml:id='code'] or
105                 ancestor::db:procedure[@xml:id='prepackage']">
106                 <xsl:value-of select="$sentinel"/>
107             </xsl:if>
108         </xsl:with-param>
109     </xsl:call-template>
110 </xsl:template>
111
112 <xsl:template match="db:sect3/db:title" mode="readme">
113     <xsl:variable name="content">
114         <xsl:apply-templates select="node()" mode="inline"/>
115     </xsl:variable>
116     <xsl:call-template name="markup">
117         <xsl:with-param name="flag" select="'####'"/>

```

```

118     </xsl:call-template>
119     <xsl:call-template name="normtext">
120       <xsl:with-param name="content"
121         select="normalize-space($content)"/>
122       <xsl:with-param name="indent">
123         <xsl:if
124           test="ancestor::db:part[@xml:id='code'] or
125             ancestor::db:procedure[@xml:id='prepackage']">
126           <xsl:value-of select="$sentinel"/>
127         </xsl:if>
128       </xsl:with-param>
129     </xsl:call-template>
130   </xsl:template>
131
132   <xsl:template match="db:bridgehead" mode="readme">
133     <xsl:variable name="content">
134       <xsl:apply-templates select="node()" mode="inline"/>
135     </xsl:variable>
136     <xsl:call-template name="markup">
137       <xsl:with-param name="flag" select="'####'"/>
138     </xsl:call-template>
139     <xsl:call-template name="normtext">
140       <xsl:with-param name="content"
141         select="normalize-space($content)"/>
142       <xsl:with-param name="indent">
143         <xsl:if
144           test="ancestor::db:part[@xml:id='code'] or
145             ancestor::db:procedure[@xml:id='prepackage']">
146           <xsl:value-of select="$sentinel"/>
147         </xsl:if>
148       </xsl:with-param>
149     </xsl:call-template>
150   </xsl:template>

```

db:warning [db:warning and db:warning/db:title]

Warning titles work the same way but with a greater-than prefix to the hash marks.

```

151   <xsl:template match="db:warning" mode="readme">
152     <xsl:if test="not(db:title)">
153       <xsl:text>&#xa;> #### WARNING</xsl:text>
154     </xsl:if>
155     <xsl:apply-templates mode="readme"/>
156     <xsl:text>&#xa;</xsl:text>
157   </xsl:template>
158
159   <xsl:template match="db:warning/db:title" mode="readme">
160     <xsl:variable name="content">
161       <xsl:apply-templates select="node()" mode="inline"/>
162     </xsl:variable>
163     <xsl:call-template name="markup">
164       <xsl:with-param name="flag" select="'&#xa;> #### '"/>
165     </xsl:call-template>

```

```

166     <xsl:call-template name="normtext">
167       <xsl:with-param name="content"
168         select="normalize-space($content)"/>
169       <xsl:with-param name="indent">
170         <xsl:if
171           test="ancestor::db:part[@xml:id='code'] or
172             ancestor::db:procedure[@xml:id='prepackage']">
173           <xsl:value-of select="$sentinel"/>
174         </xsl:if>
175       </xsl:with-param>
176     </xsl:call-template>
177   </xsl:template>

```

parawarn [db:para with [parent::db:warning or parent::db:formalpara]]

Warning paragraphs and formal paragraphs are also output with the greater-than markup.

parawarn **TODO: Not supported in code annotations (yet!)**

```

178   <!-- This has to cope with being called from
179         within db2dtx.xsl as well, to generate inline
180         documentation that passes through to
181         the .sty or .cls file -->
182   <xsl:template
183     match="db:para[parent::db:warning or
184       parent::db:formalpara]" mode="readme">
185     <xsl:variable name="content">
186       <xsl:apply-templates select="node()" mode="inline"/>
187     </xsl:variable>
188     <xsl:choose>
189       <xsl:when test="ancestor::db:chapter[@xml:id='readme']">
190         <xsl:text>> </xsl:text>
191       </xsl:when>
192       <xsl:otherwise>
193         <xsl:text>&#xa;% </xsl:text>
194       </xsl:otherwise>
195     </xsl:choose>
196     <xsl:value-of select="normalize-space($content)"/>
197     <xsl:choose>
198       <xsl:when test="ancestor::db:chapter[@xml:id='readme']">
199         <xsl:text>&#xa;> </xsl:text>
200       </xsl:when>
201       <xsl:otherwise>
202         <xsl:text>&#xa;% &#xa;</xsl:text>
203       </xsl:otherwise>
204     </xsl:choose>
205   </xsl:template>

```

db:formalpara [in db:formalpara/]

The <formalpara> structure is used in the same way as warnings.

```

206 <xsl:template match="db:formalpara/db:title" mode="readme">
207   <xsl:variable name="content">
208     <xsl:apply-templates select="node()" mode="inline"/>
209   </xsl:variable>
210   <xsl:call-template name="markup">
211     <xsl:with-param name="flag" select="''> #### '"/>
212   </xsl:call-template>
213   <xsl:call-template name="normtext">
214     <xsl:with-param name="content"
215       select="normalize-space($content)"/>
216     <xsl:with-param name="indent">
217       <xsl:if
218         test="ancestor::db:part[@xml:id='code'] or
219             ancestor::db:procedure[@xml:id='prepackage']">
220         <xsl:value-of select="$sentinel"/>
221       </xsl:if>
222     </xsl:with-param>
223   </xsl:call-template>
224   <xsl:call-template name="markup">
225     <xsl:with-param name="flag"
226       select="'' ####&#xa;> &#xa;'/>
227   </xsl:call-template>
228 </xsl:template>

```

db:programlisting We determine the type of document (class or package) so that, if the @condition is not specified, or if it matches the current document type, interpret the content rather than treating it as character data, because it may contain embedded markup like <olink>.

```

229 <xsl:template match="db:programlisting" mode="readme">
230   <xsl:variable name="curdoctype">
231     <xsl:value-of select="$maindoc/db:book/@arch"/>
232   </xsl:variable>
233   <xsl:if test="@condition=$curdoctype
234     or
235     not(@condition)">
236     <xsl:variable name="content">
237       <xsl:apply-templates select="node()"
238         mode="inline"/>
239     </xsl:variable>
240     <xsl:call-template name="markup"/>
241     <xsl:text>&#xa;</xsl:text>
242     <!-- now github method -->
243     <xsl:text>``</xsl:text>
244     <xsl:if test="@language">
245       <xsl:value-of select="@language"/>
246     </xsl:if>
247     <!-- was four leading spaces for Markdown verbatim.
248       do NOT expect to find escaped TeX in $content
249     <xsl:for-each select="tokenize($content,'&#xa;')">
250       <xsl:text>    </xsl:text>
251       <xsl:value-of select="."/>
252       <xsl:text>&#xa;</xsl:text>

```

```

253     </xsl:for-each>
254     -->
255     <xsl:value-of select="replace($content,
256                               '[ &#xa;&#x9;][ &#xa;&#x9;]*$',
257                               '&#xa;')"/>
258     <xsl:if
259       test="not(matches(substring($content,
260                               string-length($content)),
261                        '[ \n\t]'))">
262       <xsl:text>&#xa;</xsl:text>
263     </xsl:if>
264     <xsl:text>``&#xa;</xsl:text>
265   </xsl:if>
266 </xsl:template>

```

db:exceptionname Marker for RFC2119 required vocabulary.

```

267   <xsl:template match="db:exceptionname" mode="inline">
268     <xsl:value-of select="upper-case(.)"/>
269   </xsl:template>

```

db:variablelist Discussion lists (<variablelist>s) are done with the HTML <dl> element.

```

270   <xsl:template match="db:variablelist" mode="readme">
271     <xsl:call-template name="markup">
272       <xsl:with-param name="flag"
273         select="'&#xa;&lt;dl>'"/>
274     </xsl:call-template>
275     <xsl:apply-templates mode="readme"/>
276     <xsl:call-template name="markup">
277       <xsl:with-param name="flag"
278         select="'&#xa;&lt;/dl>&#xa;'"/>
279     </xsl:call-template>
280   </xsl:template>

```

db:term The <term> element in a <variablelist> becomes the <dt>.

```

281   <xsl:template match="db:term" mode="readme">
282     <xsl:variable name="content">
283       <xsl:apply-templates select="node()" mode="inline"/>
284     </xsl:variable>
285     <xsl:call-template name="markup">
286       <xsl:with-param name="flag"
287         select="'&#xa; &lt;dt>'"/>
288     </xsl:call-template>
289     <xsl:call-template name="normtext">
290       <xsl:with-param name="content"
291         select="normalize-space($content)"/>
292       <xsl:with-param name="indent">
293         <xsl:if
294           test="ancestor::db:part[@xml:id='code'] or
295                ancestor::db:procedure[@xml:id='prepackage']">
296           <xsl:value-of select="$sentinel"/>

```

```

297         </xsl:if>
298     </xsl:with-param>
299 </xsl:call-template>
300 <xsl:choose>
301     <!-- only add extra flag when outputting MD -->
302     <xsl:when test="ancestor::db:book/@arch='script'">
303         <xsl:text></xsl:text>
304     </xsl:when>
305     <xsl:otherwise>
306         <xsl:call-template name="markup">
307             <xsl:with-param name="flag" select="''&lt;/dt>'"/>
308         </xsl:call-template>
309     </xsl:otherwise>
310 </xsl:choose>
311 </xsl:template>

```

db:para [db:para with [parent::db:listitem [parent::db:orderedlist or parent::db:itemizedlist]]]

A paragraph in an itemized list or ordered list gets an asterisk for unnumbered or a digit for numbered.

```

312 <!-- inline list paragraph -->
313 <xsl:template
314     match="db:para
315         [parent::db:listitem
316         [parent::db:orderedlist
317         or
318         parent::db:itemizedlist]]" mode="readme">
319     <xsl:variable name="content">
320         <xsl:apply-templates select="node()" mode="inline"/>
321     </xsl:variable>
322     <xsl:text>&#xa;</xsl:text>
323     <xsl:call-template name="markup"/>
324     <xsl:choose>
325         <xsl:when test="ancestor::db:itemizedlist">
326             <xsl:text> * </xsl:text>
327         </xsl:when>
328         <xsl:when test="ancestor::db:orderedlist">
329             <xsl:text> </xsl:text>
330             <xsl:if
331                 test="count(parent::db:listitem/
332                     preceding-sibling::db:listitem)&lt;9">
333                 <xsl:text> </xsl:text>
334             </xsl:if>
335             <xsl:value-of
336                 select="count(parent::db:listitem/
337                     preceding-sibling::db:listitem)+1"/>
338             <xsl:text>. </xsl:text>
339         </xsl:when>
340     </xsl:choose>
341     <xsl:call-template name="normtext">
342         <xsl:with-param name="content"
343             select="normalize-space($content)"/>

```

```

344     <xsl:with-param name="indent">
345     <xsl:if
346         test="ancestor::db:part[@xml:id='code'] or
347             ancestor::db:procedure[@xml:id='prepackage']">
348         <xsl:value-of select="$sentinel"/>
349     </xsl:if>
350     </xsl:with-param>
351 </xsl:call-template>
352 <xsl:call-template name="markup"/>
353 </xsl:template>

```

db:para [db:para with [ancestor::db:variablelist]]

A paragraph in a <variablelist> item becomes a <dd>.

```

354 <xsl:template match="db:para[ancestor::db:variablelist]
355                 [not(parent::db:warning)]"
356     mode="readme">
357     <xsl:variable name="content">
358         <xsl:apply-templates select="node()" mode="inline"/>
359     </xsl:variable>
360     <xsl:call-template name="markup">
361         <xsl:with-param name="flag" select="''&#xa; &lt;dd>'"/>
362     </xsl:call-template>
363     <xsl:call-template name="normtext">
364         <xsl:with-param name="content"
365             select="normalize-space($content)"/>
366         <xsl:with-param name="indent">
367             <xsl:if
368                 test="ancestor::db:part[@xml:id='code'] or
369                     ancestor::db:procedure[@xml:id='prepackage']">
370                 <xsl:value-of select="$sentinel"/>
371             </xsl:if>
372         </xsl:with-param>
373     </xsl:call-template>
374     <xsl:choose>
375         <!-- only add extra flag when outputting MD -->
376         <xsl:when test="ancestor::db:book/@arch='script'">
377             <xsl:text></xsl:text>
378         </xsl:when>
379         <xsl:otherwise>
380             <xsl:call-template name="markup">
381                 <xsl:with-param name="flag" select="''&lt;/dd>'"/>
382             </xsl:call-template>
383         </xsl:otherwise>
384     </xsl:choose>
385 </xsl:template>

```

db:sect1 Special treatment for a <sect1> with the <ID> of "bugs".

```

386 <xsl:template match="db:sect1" mode="readme">
387     <xsl:choose>
388         <xsl:when
389             test="@xml:id='bugs' and

```

```

390         $maindoc//db:revhistory/db:revision[1]/
391         db:revdescription/db:orderedlist">
392     <xsl:apply-templates mode="readme" select="db:title"/>
393     <xsl:text>The following need attention:&#xa;&#xa;</xsl:text>
394     <xsl:apply-templates mode="readme"
395         select="$maindoc//db:revhistory/db:revision[1]/
396             db:revdescription/db:orderedlist"/>
397 </xsl:when>
398 <xsl:otherwise>
399     <xsl:apply-templates mode="readme"/>
400 </xsl:otherwise>
401 </xsl:choose>
402 <xsl:text>&#xa;</xsl:text>
403 </xsl:template>

```

db:anchor The `<anchor>` element is used to fetch the content of an element type whose name is *not* the one in `$omit` and whose ancestor is one named in the `$loc` parameter. It also gets used to retrieve the whole of the copyright statement.

db:anchor **TODO: The pos seems to be redundant now**

```

404 <xsl:template match="db:anchor" mode="readme">
405     <xsl:variable name="loc" select="@targetptr"/>
406     <xsl:variable name="pos" select="@type"/>
407     <xsl:variable name="omit" select="@remap"/>
408     <xsl:choose>
409         <!-- special usage referencing a macro -->
410         <xsl:when
411             test="@targetptr='copyright' and @type='*'">
412             <xsl:variable name="content">
413                 <xsl:for-each select="$maindoc">
414                     <xsl:call-template name="copyright-statement">
415                         <xsl:with-param name="ftype"
416                             select="/db:book/@userlevel"/>
417                     </xsl:call-template>
418                 </xsl:for-each>
419             </xsl:variable>
420             <xsl:text>&#xa;</xsl:text>
421             <xsl:for-each select="tokenize($content,'&#xa;')">
422                 <xsl:text>    </xsl:text>
423                 <xsl:value-of select="replace(.,'^','')"/>
424                 <xsl:text>&#xa;</xsl:text>
425             </xsl:for-each>
426         </xsl:when>
427         <!-- normal operation, eg abstract -->
428         <xsl:otherwise>
429             <xsl:apply-templates mode="readme"
430                 select="$maindoc/
431                     descendant::*[local-name()=$loc][1]
432                     /*[local-name()!=$omit]"/>
433         <!-- image -->
434         <xsl:if
435             test="$maindoc/

```

```

436         descendant::*[local-name()=$loc][1]
437         /@xlink:href">
438     <xsl:for-each
439         select="$maindoc/
440             descendant::*[local-name()=$loc][1]">
441         <xsl:text>&#xa;![</xsl:text>
442         <xsl:value-of select="normalize-space(@xlink:title)"/>
443         <xsl:text>]</xsl:text>
444         <xsl:if test="@vendor">
445             <xsl:value-of select="@vendor"/>
446         </xsl:if>
447         <xsl:value-of select="@xlink:href"/>
448         <xsl:text>&#xa;</xsl:text>
449     </xsl:for-each>
450 </xsl:if>
451 </xsl:otherwise>
452 </xsl:choose>
453 </xsl:template>

```

db:olink This template retrieves values from the main document via the named template
 metadata metadata.

```

454 <xsl:template match="db:olink" mode="inline">
455     <xsl:call-template name="metadata"/>
456 </xsl:template>

```

db:literallayout This is intended only for T_EX or L^AT_EX source code, outputting lines prefixed with
 four spaces.

```

457 <xsl:template match="db:literallayout" mode="inline">
458     <xsl:choose>
459         <xsl:when
460             test="@language!='LaTeX' and @language!='TeX'">
461             <!-- so this handles bash OK -->
462             <xsl:variable name="content" select="."/>
463             <xsl:call-template name="normtext">
464                 <xsl:with-param name="content"
465                     select="normalize-space($content)"/>
466                 <xsl:with-param name="indent">
467                     <xsl:if
468                         test="ancestor::db:part[@xml:id='code'] or
469                             ancestor::db:procedure[@xml:id='prepackage']">
470                         <xsl:value-of select="$sentinel"/>
471                         <xsl:text>    </xsl:text>
472                     </xsl:if>
473                 </xsl:with-param>
474             </xsl:call-template>
475             <xsl:if test="ancestor::db:part[@xml:id!='code']">
476                 <xsl:text>&#xa;</xsl:text>
477             </xsl:if>
478         </xsl:when>
479         <xsl:otherwise>
480             <xsl:text>[Raw </xsl:text>

```

```

481         <xsl:value-of select="@language"/>
482         <xsl:text> code omitted here: see PDF doc]</xsl:text>
483     </xsl:otherwise>
484 </xsl:choose>
485 </xsl:template>

```

14.2 Flow

`db:acronym` Acronyms with an `<ID>` are assumed to be defining instances with the expansion as content.

```

486 <xsl:template match="db:acronym" mode="inline">
487     <xsl:call-template name="compensate-space"/>
488     <xsl:choose>
489         <xsl:when test="@xml:id">
490             <xsl:apply-templates mode="inline"/>
491             <xsl:text> (</xsl:text>
492             <xsl:value-of select="@xml:id"/>
493             <xsl:text>)</xsl:text>
494         </xsl:when>
495         <xsl:otherwise>
496             <xsl:apply-templates mode="inline"/>
497         </xsl:otherwise>
498     </xsl:choose>
499 </xsl:template>

```

`monospace` [`db:filename` and `db:guilabel` and `db:guibutton`]

Filenames and GUI elements are rendered as monospace.

```

500 <xsl:template match="db:filename |
501                 db:guilabel |
502                 db:guibutton" mode="inline">
503     <xsl:text>`</xsl:text>
504     <xsl:apply-templates mode="inline"/>
505     <xsl:text>`</xsl:text>
506 </xsl:template>

```

italics [`db:classname` and `db:package`]

Class and package names both go in italics; this breaks the \LaTeX convention of sans-serif, but the font is not normally controllable in Markdown.

```

507 <xsl:template match="db:classname |
508                 db:package" mode="inline">
509     <xsl:text>_</xsl:text>
510     <xsl:apply-templates mode="inline"/>
511     <xsl:text>_</xsl:text>
512 </xsl:template>

```

`db:uri` URIs are rendered as-is.

```

513 <xsl:template match="db:uri" mode="inline">

```

```

514     <xsl:apply-templates mode="inline"/>
515 </xsl:template>

```

quoted [db:phrase and db:wordasword and db:quote]

Words and phrases go in double quotes.

```

516 <xsl:template match="db:phrase |
517                  db:wordasword |
518                  db:quote" mode="inline">
519   <xsl:text>"</xsl:text>
520   <xsl:apply-templates mode="inline"/>
521   <xsl:text>"</xsl:text>
522 </xsl:template>

```

db:footnote Footnotes are superscripted digits.

```

523 <xsl:template match="db:footnote" mode="inline">
524   <xsl:text>&lt;sup></xsl:text>
525   <xsl:number format="1" count="db:footnote" level="any"/>
526   <xsl:text>&lt;/sup></xsl:text>
527 </xsl:template>

```

db:xref Create textual pointers for cross-references, as there are no links as such in plain text.

```

528 <xsl:template match="db:xref" mode="inline">
529   <xsl:variable name="target"
530     select="$maindoc//*[ @xml:id=current()/@linkend ]"/>
531   <xsl:variable name="targetname"
532     select="local-name($target)"/>
533   <xsl:choose>
534     <!-- handle informatable, informalequation -->
535     <xsl:when test="starts-with($targetname,'informal')">
536       <xsl:text> the informal </xsl:text>
537       <xsl:value-of
538         select="substring-after($targetname,'informal')"/>
539       <xsl:text> labelled '</xsl:text>
540       <xsl:value-of select="@linkend"/>
541       <xsl:text>' in §</xsl:text>
542       <xsl:variable name="ref">
543         <xsl:for-each
544           select="$target/ancestor::*
545             [not(name()='part' or name()='book')]">
546           <ref n="{count(preceding-sibling::*
547             [name()='current()/name()]) + 1}">
548             <xsl:value-of select="local-name()"/>
549           </ref>
550         </xsl:for-each>
551       </xsl:variable>
552       <xsl:for-each select="$ref/ref[@n>0]">
553         <xsl:sort select="position()"
554           order="descending"/>

```

```

555         <xsl:value-of select="@n"/>
556         <xsl:if test="position()=last()">
557             <xsl:text>.</xsl:text>
558         </xsl:if>
559     </xsl:for-each>
560 </xsl:when>
561 <xsl:otherwise>
562     <xsl:text>the </xsl:text>
563     <xsl:value-of select="$targetname"/>
564 </xsl:otherwise>
565 </xsl:choose>
566 </xsl:template>

```

link Create an inline link for a URI.

```

567 <xsl:template match="db:link[@xlink:href]" mode="inline">
568     <xsl:call-template name="compensate-space"/>
569     <xsl:text>[</xsl:text>
570     <xsl:choose>
571         <xsl:when test="*">
572             <xsl:apply-templates mode="inline"/>
573         </xsl:when>
574         <xsl:otherwise>
575             <xsl:value-of select="."/>
576         </xsl:otherwise>
577     </xsl:choose>
578     <xsl:text>](</xsl:text>
579     <xsl:value-of select="replace(@xlink:href,'PACKAGE',$docname)"/>
580     <xsl:text>)</xsl:text>
581 </xsl:template>

```

db:command Prefix commands with a backslash where appropriate.

```

582 <xsl:template match="db:command" mode="inline">
583     <xsl:text>`</xsl:text>
584     <xsl:if test="@xml:lang='TeX' or
585                 @xml:lang='LaTeX' or
586                 not(@xml:lang)"">
587         <xsl:text>\</xsl:text>
588     </xsl:if>
589     <xsl:apply-templates mode="inline"/>
590     <xsl:text>`</xsl:text>
591 </xsl:template>

```

db:tag Add prefixes and suffixes to specifics of XML markup names.

```

592 <xsl:template match="db:tag" mode="inline">
593     <xsl:text>`</xsl:text>
594     <xsl:choose>
595         <xsl:when test="@class='starttag'"">
596             <xsl:text>&lt;</xsl:text>
597         </xsl:when>
598         <xsl:when test="@class='attvalue'"">

```

```

599         <xsl:text>"</xsl:text>
600     </xsl:when>
601     <xsl:when test="@class='attribute' and
602         @conformance!='noat'">
603         <xsl:text>@</xsl:text>
604     </xsl:when>
605     <xsl:when test="@class='genentity'">
606         <xsl:text>&lt;</xsl:text>
607     </xsl:when>
608     <xsl:when test="@class='numcharref'">
609         <xsl:text>&lt;\#x</xsl:text>
610     </xsl:when>
611 </xsl:choose>
612 <!-- do NOT expect to find escaped TeX here -->
613 <xsl:value-of select="."/>
614 <xsl:choose>
615     <xsl:when test="@class='starttag'">
616         <xsl:text>&gt;</xsl:text>
617     </xsl:when>
618     <xsl:when test="@class='attvalue'">
619         <xsl:text>"</xsl:text>
620     </xsl:when>
621     <xsl:when test="@class='genentity' or
622         @class='numcharref'">
623         <xsl:text>;</xsl:text>
624     </xsl:when>
625 </xsl:choose>
626 <xsl:text>`</xsl:text>
627 <!-- separate plural "s" with space,
628     otherwise rendering engines might hide it -->
629 <xsl:if
630     test="name(following-sibling::node()[1])='' and
631         starts-with(following-sibling::node()[1],'s')">
632     <xsl:text> </xsl:text>
633 </xsl:if>
634 </xsl:template>

```

`text` Surround with MD markup according to the nature of the enclosure.

```

635 <xsl:template match="text()" mode="inline">
636     <xsl:choose>
637         <xsl:when
638             test="parent::db:emphasis[@role='strong']">
639             <xsl:text>**</xsl:text>
640             <!-- do NOT expect to find escaped TeX here -->
641             <xsl:value-of select="."/>
642             <xsl:text>**</xsl:text>
643         </xsl:when>
644         <xsl:when test="parent::db:firstterm">
645             <xsl:text>*_</xsl:text>
646             <!-- do NOT expect to find escaped TeX here -->
647             <xsl:value-of select="."/>
648             <xsl:text>_*</xsl:text>
649         </xsl:when>

```

```

650     <xsl:when test="parent::db:emphasis or
651                   parent::db:productname or
652                   parent::db:citetitle">
653       <xsl:text>_</xsl:text>
654       <xsl:call-template name="dologo"/>
655       <xsl:text>_</xsl:text>
656     </xsl:when>
657     <xsl:when test="parent::db:option">
658       <xsl:text>*</xsl:text>
659       <xsl:if
660         test="parent::db:option/@language='bash'">
661         <xsl:text>-</xsl:text>
662       </xsl:if>
663       <!-- do NOT expect to find escaped TeX here -->
664       <xsl:value-of select="."/>
665       <xsl:if
666         test="parent::db:option/@language='bash'
667              and
668              not(parent::db:option/@conformance='nocolon')">
669         <xsl:text>:</xsl:text>
670       </xsl:if>
671       <xsl:text>*</xsl:text>
672     </xsl:when>
673     <xsl:when test="parent::db:varname">
674       <xsl:text>`</xsl:text>
675       <xsl:choose>
676         <xsl:when
677           test="parent::db:varname/@language='bash'">
678           <xsl:text>$</xsl:text>
679         </xsl:when>
680         <xsl:when
681           test="parent::db:varname/@language='LaTeX'">
682           <xsl:text>\</xsl:text>
683         </xsl:when>
684       </xsl:choose>
685       <!-- do NOT expect to find escaped TeX here -->
686       <xsl:value-of select="."/>
687       <xsl:text>`</xsl:text>
688     </xsl:when>
689     <xsl:otherwise>
690       <xsl:call-template name="dologo"/>
691     </xsl:otherwise>
692   </xsl:choose>
693 </xsl:template>

```

db:biblioref Create a bibliographic reference without the aid of biblatex or BibTeX.

```

694 <xsl:template match="db:biblioref" mode="inline">
695   <xsl:text>(</xsl:text>
696   <xsl:variable name="bibref" select="@linkend"/>
697   <!-- change context to the entry -->
698   <xsl:for-each
699     select="$maindoc/descendant::db:biblioentry
700           [@xml:id=$bibref]">

```

```

701     <xsl:variable name="authors"
702       select="count(descendant::db:author)"/>
703     <!-- only name up to the first two -->
704     <xsl:for-each
705       select="descendant::db:author
706         [count(preceding-sibling::db:author)&lt;2]">
707       <xsl:if test="$authors>2
708         and
709         count(preceding-sibling::db:author)=1">
710         <xsl:text>, </xsl:text>
711       </xsl:if>
712       <xsl:if
713         test="count(preceding-sibling::db:author)=2">
714         <xsl:text> & </xsl:text>
715       </xsl:if>
716       <xsl:value-of select="db:personname/db:surname"/>
717     </xsl:for-each>
718     <xsl:if test="$authors>3">
719       <xsl:text> et al</xsl:text>
720     </xsl:if>
721     <xsl:text>, </xsl:text>
722     <xsl:choose>
723       <xsl:when test="descendant::db:date">
724         <xsl:value-of
725           select="format-date
726             ((descendant::db:date)[1]/
727               @YYYY-MM-DD, '[Y]')"/>
728       </xsl:when>
729       <xsl:when test="descendant::db:confdates">
730         <xsl:value-of
731           select="format-date
732             ((descendant::db:confdates)[1]/
733               @YYYY-MM-DD-from, '[Y]')"/>
734       </xsl:when>
735       <xsl:otherwise>
736         <xsl:text>n.d.</xsl:text>
737       </xsl:otherwise>
738     </xsl:choose>
739   </xsl:for-each>
740   <xsl:text>></xsl:text>
741 </xsl:template>

```

metadata This template handles the extraction of metadata from the master file: \$loc is the name of the element; \$ctl is the [optional] name of an attribute to test; \$val is the [optional] value to test \$ctl for; \$pos is the name of the attribute to return.

```

742 <xsl:template name="metadata">
743   <xsl:variable name="loc" select="@targetptr"/>
744   <xsl:variable name="ctl" select="@condition"/>
745   <xsl:variable name="val" select="@arch"/>
746   <xsl:variable name="pos" select="@type"/>
747   <xsl:choose>
748     <!-- test an attribute -->
749     <xsl:when test="$ctl!=''">

```

```

750     <xsl:choose>
751       <!-- if a value is given -->
752       <xsl:when test="$val!=''">
753         <xsl:value-of select="$maindoc/
754           descendant::*[local-name()=$loc]
755             [@*[name()=$ctl]=$val][1]/@*[name()=$pos]"/>
756       </xsl:when>
757       <!-- if not, test for existence of $ctl attrib only -->
758       <xsl:otherwise>
759         <xsl:value-of select="$maindoc/
760           descendant::*[local-name()=$loc]
761             [@*[name()=$ctl]][1]/@*[name()=$pos]"/>
762       </xsl:otherwise>
763     </xsl:choose>
764   </xsl:when>
765   <!-- no test, just get a value -->
766   <xsl:otherwise>
767     <xsl:value-of select="$maindoc/
768       descendant::*[local-name()=$loc][1]/@*[name()=$pos]"/>
769   </xsl:otherwise>
770 </xsl:choose>
771 </xsl:template>

```

passthru [db:itemizedlist and db:listitem and db:sect2]

Don't issue warning for unmatched elements.

```

772   <xsl:template match="db:itemizedlist |
773     db:listitem |
774     db:sect2" mode="readme">
775     <xsl:apply-templates mode="readme"/>
776   </xsl:template>

```

default [*]

Warn about unhandled elements.

```

777   <xsl:template match="*" mode="readme">
778     <xsl:message>
779       <xsl:text>Unhandled readme '</xsl:text>
780       <xsl:value-of select="name()"/>
781       <xsl:text>'</xsl:text>
782     </xsl:message>
783     <xsl:apply-templates mode="readme"/>
784   </xsl:template>

```

markup Named template to output MD prefix markup context if an element is being processed. Catch the package and class containers (ie don't throw an error).

```

785   <xsl:template name="markup">
786     <xsl:param name="flag"/>
787     <xsl:choose>
788       <xsl:when
789         test="ancestor::db:book/@arch='package' or

```

```

790         ancestor::db:book/@arch='class'")
791     <xsl:text></xsl:text>
792 </xsl:when>
793 <xsl:when test="$flag!=''">
794     <xsl:text>&#xa;</xsl:text>
795     <xsl:value-of select="$flag"/>
796     <xsl:text> </xsl:text>
797 </xsl:when>
798 <xsl:when
799     test="ancestor::db:book/@arch='script'">
800     <!-- this gets done by the recursion anyway -->
801     <xsl:text></xsl:text>
802 </xsl:when>
803 <xsl:otherwise>
804     <xsl:text></xsl:text>
805 </xsl:otherwise>
806 </xsl:choose>
807 </xsl:template>

```

14.3 Typesetting plain text

`normtext` This starts the simulation the ‘typesetting’ of non-code material in plain text: this is a stub to call the setting mechanism, so we need to do any preliminary replacements of \LaTeX constructs that map to plain text now. $\$content$ is a node-set of the content to be processed; $\$indent$ is the $\$sentinel$ plus any extra space, eg for lists; $\$prefix$ is the right-hand symbol for indented material.

```

808 <xsl:template name="normtext">
809     <xsl:param name="content"/>
810     <xsl:param name="indent">
811         <xsl:text></xsl:text>
812     </xsl:param>
813     <xsl:param name="prefix">
814         <xsl:text></xsl:text>
815     </xsl:param>
816     <xsl:call-template name="set">
817         <xsl:with-param name="text"
818             select="replace(
819                 replace(
820                     replace(concat($prefix,$content),
821                         '\TeX\{\}', 'TeX'),
822                     '\LaTeX\{\}', 'LaTeX'),
823                     '\thinspace\{\}', ' '),
824                     '\nicefrac(.)().','$1/$2')"/>
826     <xsl:with-param name="indent" select="$indent"/>
827 </xsl:call-template>
828 </xsl:template>

```

`typeset` This is where the first-stage recursive work gets done. If there’s less than a line left to set, output the indent, the line, and we’re done; otherwise, grab enough

words to fill a line, output it, and try again.

```

829 <xsl:template name="set">
830   <xsl:param name="text"/>
831   <xsl:param name="indent">
832     <xsl:text></xsl:text>
833   </xsl:param>
834   <xsl:choose>
835     <!-- if the string fits the width
836          or if it's unbreakable, that's it -->
837     <xsl:when test="string-length($text) < $width
838                   or
839                   not(contains($text,' '))">
840       <xsl:value-of select="$indent"/>
841       <xsl:value-of select="$text"/>
842       <xsl:text>&#xa;</xsl:text>
843     </xsl:when>
844     <xsl:otherwise>
845       <xsl:variable name="line">
846         <xsl:call-template name="token">
847           <xsl:with-param name="text"
848             select="$text"/>
849           <xsl:with-param name="indent"
850             select="$indent"/>
851         </xsl:call-template>
852       </xsl:variable>
853       <xsl:value-of select="$line"/>
854       <xsl:text>&#xa;</xsl:text>
855       <xsl:call-template name="set">
856         <xsl:with-param name="text"
857           select="normalize-space(substring-after(
858             concat($indent,$text),$line))"/>
859         <xsl:with-param name="indent"
860           select="$indent"/>
861       </xsl:call-template>
862     </xsl:otherwise>
863   </xsl:choose>
864 </xsl:template>

```

token See what fits, a token at a time.

```

865 <xsl:template name="token">
866   <xsl:param name="text"/>
867   <xsl:param name="line">
868     <xsl:text></xsl:text>
869   </xsl:param>
870   <xsl:param name="length">
871     <xsl:text>0</xsl:text>
872   </xsl:param>
873   <xsl:param name="indent"/>
874   <xsl:param name="cycle">
875     <xsl:value-of select="number('0')"/>
876   </xsl:param>
877   <xsl:param name="context"/>

```

```

878     <xsl:variable name="word">
879       <xsl:choose>
880         <xsl:when test="contains($text,' ')">
881           <xsl:value-of
882             select="substring-before($text,' ')" />
883         </xsl:when>
884         <xsl:otherwise>
885           <xsl:value-of select="$text" />
886         </xsl:otherwise>
887       </xsl:choose>
888     </xsl:variable>
889     <xsl:variable name="size"
890       select="string-length($word)" />
891     <xsl:choose>
892       <!-- too much for line -->
893       <xsl:when test="$length + 1 + $size > $width">
894         <xsl:value-of select="$line" />
895       </xsl:when>
896       <!-- another word will fit -->
897       <xsl:otherwise>
898         <xsl:call-template name="token">
899           <xsl:with-param name="line">
900             <xsl:choose>
901               <xsl:when test="$length=0">
902                 <!-- omit indent for lists on first cycle -->
903                 <xsl:choose>
904                   <xsl:when
905                     test="($context='itemizedlist' or
906                       $context='orderedlist')
907                       and $cycle = 0">
908                     <xsl:text></xsl:text>
909                   </xsl:when>
910                   <xsl:otherwise>
911                     <xsl:value-of select="$indent" />
912                   </xsl:otherwise>
913                 </xsl:choose>
914                 <xsl:value-of select="$word" />
915               </xsl:when>
916               <xsl:otherwise>
917                 <xsl:value-of
918                   select="concat($line,' ', $word)" />
919               </xsl:otherwise>
920             </xsl:choose>
921           </xsl:with-param>
922           <xsl:with-param name="length">
923             <xsl:choose>
924               <xsl:when test="$length=0">
925                 <xsl:value-of
926                   select="string-length($indent) + $size" />
927               </xsl:when>
928               <xsl:otherwise>
929                 <xsl:value-of
930                   select="string-length($line) + 1 + $size" />
931               </xsl:otherwise>

```

```

932         </xsl:choose>
933     </xsl:with-param>
934     <xsl:with-param name="text"
935         select="substring-after($text, ' ')" />
936     <xsl:with-param name="cycle"
937         select="$cycle + 1" />
938     <xsl:with-param name="context"
939         select="$context" />
940 </xsl:call-template>
941 </xsl:otherwise>
942 </xsl:choose>
943 </xsl:template>

```

dologo Call this template when processing `<text()>` which may not be atomic, instead of doing an `<apply-templates>`. Note that these matches are for the names defined in `doctexbook.dtd`.

```

944 <xsl:template name="dologo">
945     <xsl:choose>
946         <xsl:when
947             test="contains(.,'\BibTeX{') or
948                 contains(.,'\ConTeX t{') or
949                 contains(.,'\LaTeX{') or
950                 contains(.,'\LaTeXe{') or
951                 contains(.,'\XeTeX{') or
952                 contains(.,'\XeLaTeX{') or
953                 contains(.,'\LyX{') or
954                 contains(.,'\MF{') or
955                 contains(.,'\MP{') or
956                 contains(.,'\TeX{')">
957             <xsl:call-template name="detex">
958                 <xsl:with-param name="string" select="."/>
959             </xsl:call-template>
960         </xsl:when>
961         <xsl:otherwise>
962             <xsl:value-of select="."/>
963         </xsl:otherwise>
964     </xsl:choose>
965 </xsl:template>

```

detex Perform the actual replacements of logos identified in `dologo`.

```

966 <xsl:template name="detex">
967     <xsl:param name="string"/>
968     <xsl:choose>
969         <xsl:when test="contains($string, '\BibTeX{')">
970             <xsl:call-template name="detex">
971                 <xsl:with-param name="string"
972                     select="replace($string, '\\BibTeX\\', 'BiBTeX')"/>
973             </xsl:call-template>
974         </xsl:when>
975         <xsl:when test="contains($string, '\ConTeX t{')">
976             <xsl:call-template name="detex">

```

```

977         <xsl:with-param name="string"
978           select="replace($string, '\\Con\\TeX t\\{\\}', 'ConTeXt')"/>
979       </xsl:call-template>
980   </xsl:when>
981   <xsl:when test="contains($string, '\\LaTeX{\\}')">
982       <xsl:call-template name="detex">
983           <xsl:with-param name="string"
984             select="replace($string, '\\LaTeX\\{\\}', 'LaTeX')"/>
985       </xsl:call-template>
986   </xsl:when>
987   <xsl:when test="contains($string, '\\LaTeXe{\\}')">
988       <xsl:call-template name="detex">
989           <xsl:with-param name="string"
990             select="replace($string, '\\LaTeXe\\{\\}', 'LaTeX2ε')"/>
991       </xsl:call-template>
992   </xsl:when>
993   <xsl:when test="contains($string, '\\XeTeX{\\}')">
994       <xsl:call-template name="detex">
995           <xsl:with-param name="string"
996             select="replace($string, '\\XeTeX\\{\\}', 'XeTeX')"/>
997       </xsl:call-template>
998   </xsl:when>
999   <xsl:when test="contains($string, '\\XeLaTeX{\\}')">
1000       <xsl:call-template name="detex">
1001           <xsl:with-param name="string"
1002             select="replace($string, '\\XeLaTeX\\{\\}', 'XeLaTeX')"/>
1003       </xsl:call-template>
1004   </xsl:when>
1005   <xsl:when test="contains($string, '\\LyX{\\}')">
1006       <xsl:call-template name="detex">
1007           <xsl:with-param name="string"
1008             select="replace($string, '\\LyX\\{\\}', 'LyX')"/>
1009       </xsl:call-template>
1010   </xsl:when>
1011   <xsl:when test="contains($string, '\\MF{\\}')">
1012       <xsl:call-template name="detex">
1013           <xsl:with-param name="string"
1014             select="replace($string, '\\MF\\{\\}', 'METAFONT')"/>
1015       </xsl:call-template>
1016   </xsl:when>
1017   <xsl:when test="contains($string, '\\MP{\\}')">
1018       <xsl:call-template name="detex">
1019           <xsl:with-param name="string"
1020             select="replace($string, '\\MP\\{\\}', 'METAPOST')"/>
1021       </xsl:call-template>
1022   </xsl:when>
1023   <xsl:when test="contains($string, '\\TeX{\\}')">
1024       <xsl:call-template name="detex">
1025           <xsl:with-param name="string"
1026             select="replace($string, '\\TeX\\{\\}', 'TeX')"/>
1027       </xsl:call-template>
1028   </xsl:when>
1029   <xsl:otherwise>
1030       <xsl:value-of select="$string"/>

```

```
1031         </xsl:otherwise>
1032     </xsl:choose>
1033 </xsl:template>
```

end That's all, folks!

```
1034 </xsl:stylesheet>
```

15 The Makefile

A standard GNU Makefile is supplied to handle most aspects of the production, including the packaging of the files and the zipping up for distribution to CTAN. This replaces the generated build file of earlier versions. The Makefile is fully parameterised, so the one file can be used for any package simply by soft-linking it (creating an alias) from where you installed it into the working directory you create for any class or package.

The file is extracted during installation of *ClassPack* with the name `classpack.make` to avoid any conflict with a user's existing Makefile. The file SHOULD be renamed after installation.

`usemakefile-comment` Identify the file origin, date, time, etc.

```
1035 ## The Makefile (from the classpack package v.1.28, 2024-02-21)
1036 ## Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

15.1 Auto-initialisation

This section is added automatically by *ClassPack* as a preamble to all classes and style packages. For details see the `ltxdoc` package documentation.

```
1 \NeedsTeXFormat{LaTeX2e}[2015/01/01]
2 \ProvidesFile{classpack}[2024/02/21 v1.28
3   Package code for supporting ClassPack documentation]
```

15.2 Shortcut settings

`shortcuts` As the actual filenames will be different depending on your package or class, the following shortcuts ('phony' targets in Makefile-speak) are declared:

```
2 .PHONY : all install clean tds zip ctan pdf dtx cls sty anc bib recover
3 .PHONY : test gloss ind web
4 .DEFAULT : all
```

They are defined below in [section 15.6 on page 382](#).

15.3 Document and program dependencies

`variables` The current document name `$(DOC)` is found from the name of the most recently-saved XML document in the current directory, omitting the known names of *ClassPack* support files — the result is assumed to be your class or package document. The dependencies are:

```

5 # DOC is the class/package XML document, always the most recently-edited
6 # Note: as of 2017-03-14 this does NOT need to be the same name as the dir,
7 # nor as the name of the principal class/package created
8 THISD := $(shell pwd | awk -F/ '{print $NF}')
9 DOC   := $(shell ls -1t $(THISD).xml 2>/dev/null | head -1 | cut -d . -s -f 1)
10 HERE  := $(shell pwd)
11 JAVA  := $(shell which java)
12 SAX   := $(shell locate saxon-he-10.3.jar | tail -1)
13 LXP   := $(shell which lxpprintf)
14 LMK   := $(shell which latexmk)
15 CHUNK := $(shell which chunk)

```

Java: The Sun (now Oracle) and IBM Javas as well as the generic Java supplied with GNU/Linux distributions all seem to work well;

Saxon: The HE (Home Edition) is available free of charge from <http://saxon.sourceforge.net> for domestic, hobby, academic, and pro bono use.

You are REQUIRED to purchase an EE (Enterprise Edition) license for commercial work.

lxml: is part of the LTxml suite of XML utilities from the Language Technology Group at Edinburgh University: see <https://www.ltg.ed.ac.uk/software/ltxml2/>;

latexmk: is a script distributed with T_EX Live for managing the build of complex L^AT_EX documents. It automatically handles bibliographic and index processing and is configurable for X_YL^AT_EX as well as *biber*.

15.4 Script dependencies

dependencies The *ClassPack*-specific locations are the files, scripts, and programs needed by the Makefile, and are assumed to be in one directory (here, dev within the user's Personal T_EX Directory as defined in the current T_EX installation's configuration (texmf.cnf), conventionally ~/texmf).

```

16 # ClassPack-specific locations
17 TMF := $(shell kpsewhich -expand-var '$$TEXMFHOME')
18 WEB := $(shell echo $$HOME)/Office/website/latex.silmaril.ie/packages
19 DEV := $(TMF)/dev
20 CPX := $(DEV)/db2dtx.xsl
21 CPB := $(DEV)/db2bibtex.xsl
22 CPF := $(DEV)/figtab2latex.xsl
23 CPT := $(DEV)/db2md.xsl
24 CPD := $(DEV)/decommenttbl.awk
25 CPP := $(DEV)/prepost.xml
26 #IST := $(DEV)/classpack.ist
27 IST := gind.ist
28 # from /usr/share/texlive/texmf-dist/makeindex/latex/gind.ist
29 RME := $(DEV)/readme.xml
30 LPPL := $(DEV)/lppl.xml

```

db2dtx.xsl: is the main XSLT script;

db2bibtex.xml: is xsl:included to handle the creation of a BibT_EX file;

db2md.xsl: creates the README.md file;

decommentbbl.awk: is an *awk* (1) script to handle malformed BibTeX output when using the *bibtex* processor instead of *biber*;

prepost.xml: is the lookup file of packages for auto-loading;

readme.xml: is the XML source from which the README.md file is created;

lppl.xml: is the L^AT_EX Project Public License text, used for the license of publicly-available packages and classes.

15.5 Values extracted from the document

values The *lprintf* utility is used to extract some key values from the XML document ahead of processing:

```

31 # Metadata extracted from the document
32 TYPE := $(shell $(LXP) -e book "%s" @userlevel $(DOC).xml)
33 LPROC := $(shell $(LXP) -e 'part[@xml:id="doc"]' "%s" @conformance $(DOC).xml)
34 BPROC := $(shell $(LXP) -e bibliography "%s" @condition $(DOC).xml)
35 BIBREF := $(shell $(LXP) -e biblioref "%s\n" @linkend $(DOC).xml | wc -l)
36 RFC := $(shell $(LXP) -e book "%s" 'count(/exceptionname)' $(DOC).xml)
37 RFCGOT := $(shell grep rfc2119 $(DOC).bib 2>/dev/null | wc -l)
38 VER := $(shell $(LXP) -e book "%s.%s" @version @revision $(DOC).xml)
39 #APROC := $(shell $(LXP) -e 'appendix[@arch]' "%s" 'normalize-space(@arch)' $(DOC).
xml)
40 ADOX := $(shell $(LXP) -e 'appendix[@arch]' "%s\n" @xlink:href $(DOC).xml)
41 #ADOC := $(shell $(LXP) -e 'appendix[@arch]' "%s" 'substring-before(@xlink:href
",".")' $(DOC).xml)
42 #AEXT := $(shell $(LXP) -e 'appendix[@arch]' "%s" 'substring-after(@xlink:href",".")
' $(DOC).xml)
43 DOX := $(shell $(LXP) -e appendix "%s" @xlink:href $(DOC).xml)
44 LOC := $(shell $(LXP) -e book "%s" @xml:base $(DOC).xml)

```

TYPE: the filetype (sty or cls) of the package or class being produced;

LPROC: the name of the L^AT_EX processor (*pdflatex* or *xelatex*);

BPROC: the name of the BibTeX processor (*bibtex* or *biber*);

BIBREF: The number of bibliographic citations, used to decide if we need to run the BPROC processor;

RFC: (used as a flag) the number of times the <exceptionname> element type is used in the XML document;

RFCGOT: (used as a flag) the number of times the RFC2119 bibliographic reference is actually specified in the XML document;

VER: the version number of the current XML document;

CONFIG: The name of any configuration file generated from the source;

APROC: Command needed to process an ancillary file such as a test or demo;

ADOC: The filename (without extension) of any such ancillary file;

AEXT: The filetype (extension) of any such ancillary file;

ADDSTY: The name of an additional package (.sty) file being shipped as-is with this class or package, specified in the Manifest.

15.6 Target shortcuts

shortcuts The target shortcuts allow quicker typing of the `make` command. In particular these can be used in editors and APIs that understand *make* (1) (eg in *Emacs*, pressing `F10`) and typing the shortcut will run the specified recipe).

```

45 # Target shortcuts
46 all : pdf ctan tds
47 #anc : $(ADOC).pdf
48 cls : $(DOC).$(TYPE)
49 ctan : $(DOC)-$(VER).zip
50 dtx : $(DOC).dtx
51 jpg : $(DOC).jpg
52 pdf : $(DOC).pdf
53 sty : $(DOC).$(TYPE)
54 zip : $(DOC)-$(VER).zip
55 tds : $(DOC)-$(VER).tds.zip
56 web : website
57 foo : $(TFILES)

```

Note that the `install` target is *not* included in `all`.

15.7 Transformation to .dtx and creation of the package or class

dtx The `db2dtx.xsl` script generates a number of output files in addition to the .dtx file:

```

58 # Transform the document into a .dtx file
59 $(DOC).dtx $(DOC).ins : $(DOC).xml $(CPX) $(CPB) $(CPF) $(CPT) $(CPD) $(CPP) $(RME)
60     $(JAVA) -Djdk.xml.entityExpansionLimit=0 \
61         -jar $(SAX) \
62         -xi:on \
63         -o:$@ \
64         -s:$< \
65         -dtd:on \
66         -xsl:$(CPX) \
67         cpdir=$(DEV) \
68         appdir=$(HERE)
69     $(shell echo $(RFC) exceptionnames, $(RFCGOT) entries in .bib >mflog)
70 ifeq ($(RFC),0)
71     touch $(DOC).bib
72 endif
73 # removed 2018-02-04 and added to end of db2bibtex.xsl
74 #ifeq ($(RFCGOT),0)
75 #     cat $(DEV)/rfc2119.bib >>$(DOC).bib
76 #endif
77 # Extract all the files from the .dtx
78 $(DOC).$(TYPE) : $(DOC).dtx $(DOC).ins
79     yes|xelatex $(DOC).ins

```

.dtx / .ins: the DocTeX and Installer files which are the distributable containers for the class or package;

.bib: the .bib file for the documentation bibliography, if there was one given;

MANIFEST: a MANIFEST file, used for zipping everything up for distribution;

README.md: a *Markdown* file format for CTAN to use (replacing the previous plain-text README);

Others: Any ancillary files are also created.

The \$(DOC).\$(TYPE) target (abbreviated as **cls** or **sty**) runs *pdflatex* on the .ins file to extract the class or package file and any ancillary files. The yes prefix forces them to overwrite any previous copy.

15.8 Creating the documentation

pdf The **pdf** target first checks to see if the documentation has used any of the keywords from RFC2119, and if so, adds the entry for the RFC to the .bib file.

```

80 # Create the documentation
81 $(DOC).pdf : $(DOC).dtx $(DOC).$(TYPE) $(DOC).bib $(CPX) $(CPB) $(CPT) $(CPD) $(CPP)
82 # XeLaTeX is required
83     $(LMK) -xelatex -bibtex $<
84     makeindex -s gglo.ist -o $(DOC).gls $(DOC).glo
85     makeindex -s gind.ist -o $(DOC).ind $(DOC).idx
86     $(LMK) -xelatex -bibtex $<
87 #     sed -e "s+  {+{+" $(DOC).idx >$(DOC).newidx
88
89 # Alternative for one-shot setting
90
91 test : $(DOC).dtx $(DOC).$(TYPE) $(DOC).bib $(CPX) $(CPB) $(CPT) $(CPD) $(CPP)
92     $(LPROC) $(DOC).dtx
93
94 $(DOC).jpg : $(DOC).pdf
95     convert -density 300 $<[0] -quality 100 -flatten -sharpen 0x1.0 $@
96
97 # TODO: add processing demo/test file[s]
```

It then executes either *pdflatex* or XeLaTeX to create the documentation. The *latexmk* script is not currently used, because of recurrent problems it has in detecting a good build. In this version of *ClassPack*, using XeLaTeX assumes the use of *biber*, and the use of *pdflatex* assumes the use of *bibtex*.

The use of the `decommentbbl.awk` script on the output from *bibtex* is to defeat the use of terminal percent comment characters in some BIBTEX styles, which upset the *ltxdoc* package because of the special use of that character there.

The *sed*(1) utility is used to remove the two intrusive spaces in index entry formatting caused by premature ingestion of the index argument in restricted mode in LaTeX which would otherwise cause multiple entries for the same index term.

15.9 Package the files for CTAN

ctan The plain (CTAN) Zip file just copies the files listed in the MANIFEST into a directory named after the class or package, removes any generated files (those from chapters in the <files> Part which will get recreated), and zips the rest up flat.

```

98 # Package as plain zip file for CTAN: top level directory only
99 $(DOC)-$(VER).zip : README.md MANIFEST $(DOC).pdf $(DOC).dtx $(DOC).ins Makefile
100     /bin/rm -rf $(DOC)/
101     mkdir -p $(DOC)
102     $(LXP) -e '/book/info/revhistory/revision[not(../revision/@version > @version)]' \
103         "Date of most recent change is %s\n" date/@YYYY-MM-DD $(DOC).xml >VERSION
104 #     unzip -v $(DOC) | grep -E '[12][0-9]{3}-[01][0-9]-[012][0-9] [0-2][0-9]:[0-5][0-9]' | awk '{
105     print "Date of most-recently-changed file is " $$5 "T" $$6 ":00"}' | sort | tail -1 > VERSION
106     cp -f `cat MANIFEST | grep -Ev '(\.dtd|\.xml|\.sh|\.awk|Makefile|MANIFEST)$$'` $(DOC)/
107     rm -f `lxprintf -e 'part[@xml:id="files"]/chapter' "$(DOC)/%s\n" @xlink:href $(DOC).
108     xml`
109     cp -f $(DOC).dtx $(DOC).ins $(DOC)/ 2>>zip.log
110     /bin/rm -f $(DOC)
111     zip -r -FS -T \
112         --exclude=*.svn* --exclude=*.DS_Store* $(DOC)
113     cp VERSION $(DOC)/VERSION
114     ls -1 $(DOC) > $(DOC)/MANIFEST
115     zip $(DOC) $(DOC)/VERSION $(DOC)/MANIFEST

```

As recommended by CTAN, if the zipped file or its results are likely to contain ancillary files, images, or other files as well as the standard DocTeX files, a standalone VERSION file should give the timestamp of the most recently-updated file in the bundle. This is not currently implemented for the TDS zip file, only for the CTAN one.

15.10 Package the files as TDS

tds The final stage is to create the Zip files of the class or package. The TDS version is created by extracting from the Manifest the filenames by type of file which need to be copied into the relevant three subdirectories named after the package or class in doc/latex, source/latex, and tex/latex, and zipping them up recursively to preserve the tree.

```

114 # Package as TDS zip file.
115 # Clear out the doc|source|tex|generic directories
116 # then populate them from the MANIFEST
117 # TODO: stop packaging generated files
118 $(DOC)-$(VER).tds.zip : README.md MANIFEST $(DOC).pdf \
119     $(DOC).dtx $(DOC).ins $(DOC).$(TYPE) Makefile
120     /bin/rm -rf doc/* source/* tex/* source/$(DOC) 2>>$(DOC).err
121     mkdir -p doc/latex/$(DOC)
122     mkdir -p source/latex/$(DOC)
123     mkdir -p $(LOC)/$(DOC)
124     mkdir -p tex/generic/$(DOC)
125     cp -f README.md MANIFEST $(DOC).pdf doc/latex/$(DOC)/
126     cp -f `cat MANIFEST | grep -E '\.(dtd|lawk|sh|xml|dtx|tex|bib|csl|ins)$$'` source/
127     latex/$(DOC) 2>>zip.log
128     if [ `ls -1 *.cls 2>/dev/null | wc -l` -gt 0 ]; then cp -f *.cls $(LOC)/$(DOC); fi
129     if [ `ls -1 *.sty 2>/dev/null | wc -l` -gt 0 ]; then cp -f *.sty $(LOC)/$(DOC); fi
130     if [ `ls -1 *.cnf 2>/dev/null | wc -l` -gt 0 ]; then cp -f *.cnf $(LOC)/$(DOC); fi

```

```

130     if [ `ls -1 *.def 2>/dev/null | wc -l` -gt 0 ]; then cp -f *.def $(LOC)/$(DOC); fi
131     if [ `cat MANIFEST | grep -E '\.(jpg|png|pdf|make)$' | grep -v $(DOC).pdf | wc -l` -gt
    0 ]; then cp -f `cat MANIFEST | grep -E '\.(jpg|png|pdf|make)$' | grep -v $(DOC).pdf` tex
    /generic/$(DOC) 2>>zip.log; fi
132 ifeq ($(ADOC),)
133 else
134     cp -f $(ADOX) tex/latex/$(DOC)/
135 endif
136 ifeq ($(DOC),classpack)
137     mkdir -p dev/$(DOC)
138     for f in awk sh xsl xml dtd; do if [ -f source/latex/$(DOC)/*.$f ]; then mv -f source/
    latex/$(DOC)/*.$f dev/$(DOC); fi; done
139 endif
140     /bin/rm -f $@
141     zip -r -FS -T \
142         --exclude=*.svn* --exclude=*.DS_Store* $@ \
143         doc/latex/$(DOC) source/latex/$(DOC) $(LOC)/$(DOC) tex/generic/* 2>>$(DOC).err

```

Because there could be both `.cls` and `.sty` files in one package, and because *make* would pick up on an error of one type not existing in a wildcard copy operation, each one has to be set inside a condition that tests for that type of file.

15.11 Chunking and fragmenting

Long XML or XSLT files reproduced with the listings package using the starting and ending line number options for chunking cause an unusable slow-down in processing, as the entire file has to be re-read from the start each time, and listings currently has a reported bug which introduces bogus characters before each listing.

The temporary solution is to insert flagged string values in the form of comments to delimit the chunks as listed in the table on page 62, and use the strings as start and end delimiters as described in the listings package documentation. This means the large files have to be preprocessed to split them into the delimited chunks with a recognisable filename pattern. It is also important to ensure that the number of chunks matches the number of programlisting elements needed to list those fragments included in your documentation.

The *chunk* (1) shell script and its associated *chunk.awk* script do the splitting, and also create the *DocBook* code for each fragment.

fragments We can check that the list of files denoted as fragments in the author's *ClassPack* document code matches the list of fragment files generated by the *chunk* (1) script.

```

143 # Find dependencies in document and [re-]chunk them
144 fragcheck : $(DOC).xml $(CHUNK)
145     $(LXP) -e 'programlisting[@conformance="frag"]' "%s\n" @xlink:href $< | \
146         sort | uniq >fragfiles.list
147     ls -1 *.frag | awk -F- '{print $$1}' | sort | uniq >fragdocs.list
148     join -j 1 -t . -a 1 fragfiles.list fragdocs.list
149     for f in `cat fragfiles.list`; do $(CHUNK) -p cpdoc $$f; done
150     cp Makefile Makefile.make

```

15.12 Installations

There are two installations done here.

15.12.1 System installation

The system installation is for normal use of the package or class in the user's \LaTeX work.

`install` This performs a local **install** on the user's system by unzipping the TDS Zip file into the user's Personal \TeX Directory.

```

152 # Install locally, will remake .tds.zip if needed
153 install : $(DOC)-$(VER).tds.zip $(DOC)-$(VER).zip
154         unzip -o -d $(TMF) $(DOC)-$(VER).tds.zip
155 #         svn add $(TMF)/doc/latex/$(LOC)
156 #         svn add $(TMF)/source/latex/$(LOC)
157 #         svn add $(TMF)/tex/latex/$(LOC)
158 #         svn add $(TMF)/tex/generic/
159 #         svn ci -m $(DOC)-install
160
161 ## web site for LaTeX packages ##
162 # Copy locally; FTP to the live site later
163 # Add directory to svn the first time!

```

15.12.2 Web installation

The web installation is principally for the user's own web site, replicated on the local disk.

`web` This installs the package in a directory which is part of a local *Apache* or other web server product.

```

165 website : $(DOC)-$(VER).tds.zip
166         mkdir -p $(WEB)/$(DOC)
167         cp index.html $(WEB)/$(DOC)
168         unzip -o -j -d $(WEB)/$(DOC) $<
169         cp -f `cat MANIFEST` $(WEB)/$(DOC)

```

15.13 Recovery and utilities

`recover` The **recover** target can be used when \LaTeX has got itself wedged over an unexpected error type: it performs a full run plus a run of the bibliography.

```

171 # Recover from fatal LaTeX or Bib or index error after fixing
172 recover : $(DOC).dtx
173         $(LPROC) --interaction=nonstopmode $(DOC).dtx
174         $(BPROC) $(DOC)
175 gloss : $(DOC).dtx $(DOC).glo
176         makeindex -s gglo.ist -o $(DOC).gls $(DOC).glo
177 ind : $(DOC).dtx $(DOC).idx
178         makeindex -s gind.ist -o $(DOC).ind $(DOC).idx
179 bib : $(DOC).dtx $(DOC).bib
180         $(BPROC) $(DOC)
181 anew : $(DOC).dtx

```

```
182 $(LMK) -gg --interaction=nonstopmode -xelatex -bibtex $(DOC).dtx
```

The additional targets recreate the glossaries, indexes, and bibliography; and reprocess the .dtx file.

`clean` The **clean** target deletes all created files for the current version *excluding* (obviously) the XML master file itself and earlier zip files.

```
183  
184 #TFILES= $(shell cat fragfiles.list)  
185 TFILES := $(file < fragfiles.list)  
186 $(TFILES): fragfiles.list  
187     head $(DOC).xml
```

16 The `decommentbb1.awk` script

`decommentscript-comment` Identify the file origin, date, time, etc.

```
1 ## The decommentbb1.awk script (from the classpack package v.1.28, 2024-02-21)
2 ## Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

The `decommentbb1.awk` script is used by the Makefile on the output from *bibtex* (only, not *biblatex*) to defeat the use of terminal percent comment characters in some \LaTeX styles, which upsets the *ltxdoc* package because of the special use of that character there.

This is an *awk* (1) script to catenate consecutive lines of a \LaTeX -produced `.bb1` file which end in a comment character (removing the comment character in the process) because the default `%` comment character gets misinterpreted by the *ltxdoc* package when building class or package files.

`decommentbb1` Read the whole file as lines, then catenate any line ending in a percent sign to the previous one.

```
3 #
4 #####
5 #
6 # Read every line of the file into an array
7
8 {
9     line[NR]=$0;
10 }
11
12 #####
13 #
14 # At the end, go through the array; if a line ends with
15 # a % sign, catenate it to the buffer, omitting the percent
16 # sign itself; otherwise, output the buffer and zero it for
17 # further use.
18
19 END {
20     for(i=1;i<=NR;++i) {
21         if(substr(line[i],length(line[i]))=="%") {
22             buffer=buffer substr(line[i],1,length(line[i])-1);
23         } else {
24             print buffer line[i];buffer="";
25         }
26     }
27 }
```

Otherwise, output any accumulated line, lather, rinse, repeat.

17 The chunk.sh shell script

This shell script extracts all the named Processing Instructions in an XML document (including XSLT) and creates a set of <annotation> templates for documentation.

17.1 Script code

chunk There is one MANDATORY switch, **-p**, which takes an argument of the Processing Instruction (PI) name; in *ClassPack* this is *cpdoc*.

This MUST be followed by the name of the document to process.

Extraction and formatting is done in the *awk* (1) script *chunk.awk* (see [section 18 on page 393](#)).

```
1  #! /bin/bash
```

chunkscrip-script-comment Identify the file origin, date, time, etc.

```
2  ## The chunk.sh shell script (from the classpack package v.1.28, 2024-02-21)
3  ## Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

```
4  # Bourne shell script to create chunks of a long document
5  # splitting on named boundaries in [doubled] comment characters
6  # or (XML) Processing Instructions. See §6.3.4 of Classpack PDF
7
8  # 1.3 added dbincludes to assist post-editing
9
10 VERSION=1.3
11 VDATE="2024-02-19"
12 DEV=~ /texmf/dev
13 DIR=`pwd`
14 APP=`basename $DIR`/`basename $DIR`.xml
15 SCRIPT=~ /bin/chunk.awk
16
17 if [ -z "$1" ]; then
18     cat <<EOF
19 Synopsis
20
21 $ chunk -p name file
22
23 -p specifies a PI name to use for splitting XML (no default)
24
25 file name of the XML document from dev to process
26
27 Output files are named after the values of the chunk boundaries
28 eg filename-xxxxxx.ext
29 EOF
30     exit 0
```

```

31 else
32     while [[ $# -gt 0 ]]; do
33     key="$1"
34     case $key in
35         -p)
36         BOUNDARY="$2"
37         shift
38         shift
39         ;;
40         *)
41         if [ ! -s "$DEV/$key" ]; then
42             echo No such file as "$DEV/$key"
43             exit 1
44         else
45             INPUT="$DEV/$key"
46             EXT=`echo $INPUT | awk -F. '{print $NF}'`
47             FNAME=`basename $INPUT .$EXT`
48             if [ -z "$BOUNDARY" ]; then
49             if [ "$EXT" = "tex" ]; then
50                 CHUNKS=`grep '^%% [A-Za-z][A-Za-z\~]* %%' $INPUT | \
51                 wc -l`
52                 BOUNDARY="%%"
53             elif [ "$EXT" = "css" ]; then
54                 CHUNKS=`grep '^/* [A-Za-z][A-Za-z\~]* */' $INPUT | \
55                 wc -l`
56                 BOUNDARY="/* */"
57             elif [ "$EXT" = "awk" -o "$EXT" = "make" ]; then
58                 CHUNKS=`grep '^## [A-Za-z][A-Za-z\~]* ##' $INPUT | \
59                 wc -l`
60                 BOUNDARY="##"
61             else # scripts with no filetype
62                 CHUNKS=`grep '^## [A-Za-z][A-Za-z\~]* ##' $INPUT | \
63                 wc -l`
64                 BOUNDARY="##"
65             fi
66             else
67             if [ "$EXT" = "xml" -o "$EXT" = "xsl" -o "$EXT" = "dtd" ]; then
68                 CHUNKS=`grep '^<?$BOUNDARY [A-Za-z][A-Za-z\~]*[ ]*>' $INPUT | wc -l`
69             else
70                 echo The -p PI boundary cannot be used with non-XML/XSLT files
71                 exit 3
72             fi
73             fi
74             if [ "$CHUNKS" -eq 0 ]; then
75                 echo No $BOUNDARY fragments found in $INPUT
76                 echo Nothing to do
77                 exit 4
78             else
79                 echo $CHUNKS $BOUNDARY fragments found in $INPUT
80             fi
81         fi
82     shift
83     ;;
84 esac

```

```

85     done
86 fi
87
88 cat <<EOF
89 This is fragment chunk, v.$VERSION ($VDATE) by Silmaril for ClassPack
90 Read from $INPUT, chunking on $CHUNKS $BOUNDARY boundaries
91 EOF
92
93 cat <<EOF >$FNAME.chunks
94 <chunks source="$INPUT" date="`date --iso-8601=seconds`"
95     boundary="$BOUNDARY" xmlns:xlink="http://www.w3.org/1999/xlink">
96 EOF
97
98 cat $INPUT | awk -v boundary="$BOUNDARY" \
99     -v fname="$FNAME" \
100     -v ext="$EXT" \
101     -f $SCRIPT
102
103 LINES=`grep -v $BOUNDARY $INPUT | wc -l`
104 echo $LINES "lines of code and comment"
105
106 cat <<EOF >>$FNAME.chunks
107 </chunks>
108 EOF
109
110 # Summaries
111
112 lxprintf -e programlisting[@xlink:href="$3"][@conformance="frag"] \
113     "%s\n" @startinglinenumber $DEV/$APP >$FNAME.doclist
114
115 lxprintf -e programlisting[@xlink:href="$3"][@conformance="frag"] \
116     "%s\n" @startinglinenumber $FNAME.chunks >$FNAME.actual
117
118 COMP=`diff $FNAME.doclist $FNAME.actual`
119
120 cat <<EOF >$FNAME-dbincludes.xml
121 <?xml version="1.0" encoding="UTF-8"?>
122 <!DOCTYPE sect1
123     PUBLIC "+//DTD Silmaril//DocBook 5 adapted for ClassPack//EN"
124     "../doctexbook-master.dtd">
125     <sect1>
126         <title></title>
127 EOF
128 cat $FNAME.actual | while read CHUNKNAME; do
129
130     cat <<EOF >>$FNAME-dbincludes.xml
131         <annotation role="" xreflabel="$FNAME">
132             <para></para>
133             <programlisting xlink:href="$3"
134                 language="XML" conformance="frag"
135                 startinglinenumber="$CHUNKNAME"/>
136         </annotation>
137 EOF
138

```

```

139 done
140 cat <<EOF >>$FNAME-dbincludes.xml
141     </sect1>
142 EOF
143
144 if [ -n "$COMP" ]; then
145     echo $DEV/$APP and $FNAME.xml have different chunk references
146     exit 5
147 fi
148
149 exit

```

Output is created in a file with the same name as the input file but with the file-type `.chunks`, so for the command `$./chunk -p cpdoc db2bibtex.xml` the following output is created:

```

<chunks source="db2bibtex.xml" date="2019-12-17T22:41:48+00:00">
  <annotation role="template" arch="paragraph" xreflabel="init">
    <para></para>
    <programlisting xlink:href="db2bibtex.xml"
      language="XSLT" conformance="frag"
      startinglinenumber="init"/>
  </annotation>
  ...
</chunks>

```

18 The chunk.awk script

chunkawkscript-comment Identify the file origin, date, time, etc.

```
1 ## The chunk.awk script (from the classpack package v.1.28, 2024-02-21)
2 ## Generated by ClassPack v.1.28 (2024-02-21) on 24-02-26T13:47:16
```

These lines are automatically prepended to the file and may be removed if necessary, using a suitable editor.

chunkawk This script processes an XML or XSLT file and finds all the Processing Instructions. They must occur in start/finish pairs, delimiting the code to which they refer, so the script assumes that the even-numbered ones are the ends of chunks. For each completed chunk, the code gets output to a file named after the main file, with the name of the PI appended.

```
3 BEGIN {
4     fn="";
5     chunks=0;
6     lines=0;
7     totlines=0;
8 }
9
10 # Match a boundary starting a line (optional leading TABs or spaces)
11
12 /^[ \t]*<[^\>]*\?>/ {
13     n=split($0,t,"?");
14     m=split(t[2],p," ");
15     if (p[1]==boundary) {
16         ++pi;
17         if (pi/2==int(pi/2)) {
18             print fn,lines,"lines";
19             totlines=totlines+lines;
20             lines=0;
21             fn="";
22         } else {
23             fn=fname "-" p[2] ".frag";
24             just=0;
25             ++chunks;
26             if (p[2] in cname) {
27                 print "Chunk name",p[2],"already used"
28                 exit 1
29             } else {
30                 cname[p[2]]=chunks;
31             }
32             file=fname ".chunks";
33             print " <annotation" >>file;
34             print "     role=\"template\" >>file;
35             print "     arch=\"paragraph\" >>file;
36             print "     xreflabel=\"\" p[2] \"\" >>file;
37             print " <para></para> >>file;
38             print " <programlisting\" >>file;
39             print "     xlink:href=\"\" fname "." ext "\" >>file;
```

```

40         print "      language=\"XSLT\" >>file;
41         print "      conformance=\"frag\" >>file;
42         print "      startinglinenumber=\"\" p[2] \"\"/>\" >>file;
43         print "    </annotation>\" >>file;
44     }
45 }
46 }
47
48 {
49     if (fn!=""&&just>0) {
50         ++lines;
51     print $0 >fn;
52     }
53 }
54
55 {
56     just=NR;
57 }
58
59 END {
60     print chunks,"chunks in",totlines,"lines";
61 }

```

19 The \LaTeX Project Public License (v 1.3c)

Everyone is allowed to distribute verbatim copies of this license document, but modification of it is not allowed.

19.1 Preamble

The \LaTeX Project Public License (LPPL) is the primary license under which the \LaTeX kernel and the base \LaTeX packages are distributed.

You may use this license for any work of which you hold the copyright and which you wish to distribute. This license may be particularly suitable if your work is \TeX -related (such as a \LaTeX package), but it is written in such a way that you can use it even if your work is unrelated to \TeX .

The section “Whether and How to Distribute Works under This License”, below, gives instructions, examples, and recommendations for authors who are considering distributing their works under this license.

This license gives conditions under which a work may be distributed and modified, as well as conditions under which modified versions of that work may be distributed.

We, the \LaTeX 3 Project, believe that the conditions below give you the freedom to make and distribute modified versions of your work that conform with whatever technical specifications you wish while maintaining the availability, integrity, and reliability of that work. If you do not see how to achieve your goal while meeting

these conditions, then read the document `cfgguide.tex` and `modguide.tex` in the base \LaTeX distribution for suggestions.

19.2 Definitions

In this license document the following terms are used:

Work: Any work being distributed under this License.

Derived Work: Any work that under any applicable law is derived from the Work.

Modification: Any procedure that produces a Derived Work under any applicable law — for example, the production of a file containing an original file associated with the Work or a significant portion of such a file, either verbatim or with modifications and/or translated into another language.

Modify: To apply any procedure that produces a Derived Work under any applicable law.

Distribution: Making copies of the Work available from one person to another, in whole or in part. Distribution includes (but is not limited to) making any electronic components of the Work accessible by file transfer protocols such as FTP or HTTP or by shared file systems such as Sun’s Network File System (NFS).

Compiled Work: A version of the Work that has been processed into a form where it is directly usable on a computer system. This processing may include using installation facilities provided by the Work, transformations of the Work, copying of components of the Work, or other activities. Note that modification of any installation facilities provided by the Work constitutes modification of the Work.

Current Maintainer: A person or persons nominated as such within the Work. If there is no such explicit nomination then it is the ‘Copyright Holder’ under any applicable law.

Base Interpreter: A program or process that is normally needed for running or interpreting a part or the whole of the Work.

A Base Interpreter may depend on external components but these are not considered part of the Base Interpreter provided that each external component clearly identifies itself whenever it is used interactively. Unless explicitly specified when applying the license to the Work, the only applicable Base Interpreter is a ‘ \LaTeX -Format’ or in the case of files belonging to the ‘ \LaTeX -format’ a program implementing the ‘ \TeX language’.

19.3 Conditions on Distribution and Modification

1. Activities other than distribution and/or modification of the Work are not covered by this license; they are outside its scope. In particular, the act of

running the Work is not restricted and no requirements are made concerning any offers of support for the Work.

2. You may distribute a complete, unmodified copy of the Work as you received it. Distribution of only part of the Work is considered modification of the Work, and no right to distribute such a Derived Work may be assumed under the terms of this clause.
3. You may distribute a Compiled Work that has been generated from a complete, unmodified copy of the Work as distributed under Clause 2 above, as long as that Compiled Work is distributed in such a way that the recipients may install the Compiled Work on their system exactly as it would have been installed if they generated a Compiled Work directly from the Work.
4. If you are the Current Maintainer of the Work, you may, without restriction, modify the Work, thus creating a Derived Work. You may also distribute the Derived Work without restriction, including Compiled Works generated from the Derived Work. Derived Works distributed in this manner by the Current Maintainer are considered to be updated versions of the Work.
5. If you are not the Current Maintainer of the Work, you may modify your copy of the Work, thus creating a Derived Work based on the Work, and compile this Derived Work, thus creating a Compiled Work based on the Derived Work.
6. If you are not the Current Maintainer of the Work, you may distribute a Derived Work provided the following conditions are met for every component of the Work unless that component clearly states in the copyright notice that it is exempt from that condition. Only the Current Maintainer is allowed to add such statements of exemption to a component of the Work.
 - (a) If a component of this Derived Work can be a direct replacement for a component of the Work when that component is used with the Base Interpreter, then, wherever this component of the Work identifies itself to the user when used interactively with that Base Interpreter, the replacement component of this Derived Work clearly and unambiguously identifies itself as a modified version of this component to the user when used interactively with that Base Interpreter.
 - (b) Every component of the Derived Work contains prominent notices detailing the nature of the changes to that component, or a prominent reference to another file that is distributed as part of the Derived Work and that contains a complete and accurate log of the changes.
 - (c) No information in the Derived Work implies that any persons, including (but not limited to) the authors of the original version of the Work, provide any support, including (but not limited to) the reporting and handling of errors, to recipients of the Derived Work unless those persons have stated explicitly that they do provide such support for the Derived Work.
 - (d) You distribute at least one of the following with the Derived Work:

- i. A complete, unmodified copy of the Work; if your distribution of a modified component is made by offering access to copy the modified component from a designated place, then offering equivalent access to copy the Work from the same or some similar place meets this condition, even though third parties are not compelled to copy the Work along with the modified component;
 - ii. Information that is sufficient to obtain a complete, unmodified copy of the Work.
- 7. If you are not the Current Maintainer of the Work, you may distribute a Compiled Work generated from a Derived Work, as long as the Derived Work is distributed to all recipients of the Compiled Work, and as long as the conditions of Clause 6 on the previous page, above, are met with regard to the Derived Work.
- 8. The conditions above are not intended to prohibit, and hence do not apply to, the modification, by any method, of any component so that it becomes identical to an updated version of that component of the Work as it is distributed by the Current Maintainer under Clause 4 on the preceding page, above.
- 9. Distribution of the Work or any Derived Work in an alternative format, where the Work or that Derived Work (in whole or in part) is then produced by applying some process to that format, does not relax or nullify any sections of this license as they pertain to the results of applying that process.
- 10. (a) A Derived Work may be distributed under a different license provided that license itself honors the conditions listed in Clause 6 on the previous page above, in regard to the Work, though it does not have to honor the rest of the conditions in this license.
- (b) If a Derived Work is distributed under a different license, that Derived Work must provide sufficient documentation as part of itself to allow each recipient of that Derived Work to honor the restrictions in Clause 6 on the preceding page above, concerning changes from the Work.
- 11. This license places no restrictions on works that are unrelated to the Work, nor does this license place any restrictions on aggregating such works with the Work by any means.
- 12. Nothing in this license is intended to, or may be used to, prevent complete compliance by all parties with all applicable laws.

19.4 No Warranty

There is no warranty for the Work. Except when otherwise stated in writing, the Copyright Holder provides the Work ‘as is’, without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the Work is with you. Should the Work prove defective, you assume the cost of all necessary servicing, repair, or correction.

In no event unless required by applicable law or agreed to in writing will The Copyright Holder, or any author named in the components of the Work, or any other party who may distribute and/or modify the Work as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of any use of the Work or out of inability to use the Work (including, but not limited to, loss of data, data being rendered inaccurate, or losses sustained by anyone as a result of any failure of the Work to operate with any other programs), even if the Copyright Holder or said author or said other party has been advised of the possibility of such damages.

19.5 Maintenance of The Work

The Work has the status ‘author-maintained’ if the Copyright Holder explicitly and prominently states near the primary copyright notice in the Work that the Work can only be maintained by the Copyright Holder or simply that it is ‘author-maintained’.

The Work has the status ‘maintained’ if there is a Current Maintainer who has indicated in the Work that they are willing to receive error reports for the Work (for example, by supplying a valid e-mail address). It is not required for the Current Maintainer to acknowledge or act upon these error reports.

The Work changes from status ‘maintained’ to ‘unmaintained’ if there is no Current Maintainer, or the person stated to be Current Maintainer of the work cannot be reached through the indicated means of communication for a period of six months, and there are no other significant signs of active maintenance.

You can become the Current Maintainer of the Work by agreement with any existing Current Maintainer to take over this role.

If the Work is unmaintained, you can become the Current Maintainer of the Work through the following steps:

1. Make a reasonable attempt to trace the Current Maintainer (and the Copyright Holder, if the two differ) through the means of an Internet or similar search.
2. If this search is successful, then enquire whether the Work is still maintained.
 - (a) If it is being maintained, then ask the Current Maintainer to update their communication data within one month.
 - (b) If the search is unsuccessful or no action to resume active maintenance is taken by the Current Maintainer, then announce within the pertinent community your intention to take over maintenance. (If the Work is a \LaTeX work, this could be done, for example, by posting to [news:comp.text.tex](https://www.tex.ac.uk/latex/news:comp.text.tex)).
3. (a) If the Current Maintainer is reachable and agrees to pass maintenance of the Work to you, then this takes effect immediately upon announcement.
 - (b) If the Current Maintainer is not reachable and the Copyright Holder agrees that maintenance of the Work be passed to you, then this takes

effect immediately upon announcement.

4. If you make an ‘intention announcement’ as described in [2b on the previous page](#) above and after three months your intention is challenged neither by the Current Maintainer nor by the Copyright Holder nor by other people, then you may arrange for the Work to be changed so as to name you as the (new) Current Maintainer.
5. If the previously unreachable Current Maintainer becomes reachable once more within three months of a change completed under the terms of [3b on the preceding page](#) or 4, then that Current Maintainer must become or remain the Current Maintainer upon request provided they then update their communication data within one month.

A change in the Current Maintainer does not, of itself, alter the fact that the Work is distributed under the LPPL license.

If you become the Current Maintainer of the Work, you should immediately provide, within the Work, a prominent and unambiguous statement of your status as Current Maintainer. You should also announce your new status to the same pertinent community as in [2b on the preceding page](#) above.

19.6 Whether and How to Distribute Works under This License

This section contains important instructions, examples, and recommendations for authors who are considering distributing their works under this license. These authors are addressed as ‘you’ in this section.

19.6.1 Choosing This License or Another License

If for any part of your work you want or need to use *distribution* conditions that differ significantly from those in this license, then do not refer to this license anywhere in your work but, instead, distribute your work under a different license. You may use the text of this license as a model for your own license, but your license should not refer to the LPPL or otherwise give the impression that your work is distributed under the LPPL.

The document `modguide.tex` in the base \LaTeX distribution explains the motivation behind the conditions of this license. It explains, for example, why distributing \LaTeX under the GNU General Public License (GPL) was considered inappropriate. Even if your work is unrelated to \LaTeX , the discussion in `modguide.tex` may still be relevant, and authors intending to distribute their works under any license are encouraged to read it.

19.6.2 A Recommendation on Modification Without Distribution

It is wise never to modify a component of the Work, even for your own personal use, without also meeting the above conditions for distributing the modified component. While you might intend that such modifications will never be distributed, often this will happen by accident — you may forget that you have modified that component; or it may not occur to you when allowing others to access the modified version that you are thus distributing it and violating the conditions of this

license in ways that could have legal implications and, worse, cause problems for the community. It is therefore usually in your best interest to keep your copy of the Work identical with the public one. Many works provide ways to control the behavior of that work without altering any of its licensed components.

19.6.3 How to Use This License

To use this license, place in each of the components of your work both an explicit copyright notice including your name and the year the work was authored and/or last substantially modified. Include also a statement that the distribution and/or modification of that component is constrained by the conditions in this license.

Here is an example of such a notice and statement:

```
%%% pig.dtx
%%% Copyright 2005 M. Y. Name
%%%
%% This work may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either version 1.3
%% of this license or (at your option) any later version.
%% The latest version of this license is in
%%   http://www.latex-project.org/lppl.txt
%% and version 1.3 or later is part of all distributions of LaTeX
%% version 2005/12/01 or later.
%%
%% This work has the LPPL maintenance status `maintained'.
%%
%% The Current Maintainer of this work is M. Y. Name.
%%
%% This work consists of the files pig.dtx and pig.ins
%% and the derived file pig.sty.
```

Given such a notice and statement in a file, the conditions given in this license document would apply, with the ‘Work’ referring to the three files `pig.dtx`, `pig.ins`, and `pig.sty` (the last being generated from `pig.dtx` using `pig.ins`), the ‘Base Interpreter’ referring to any ‘ \LaTeX -Format’, and both ‘Copyright Holder’ and ‘Current Maintainer’ referring to the person M. Y. Name.

If you do not want the Maintenance section of LPPL to apply to your Work, change ‘maintained’ above into ‘author-maintained’. However, we recommend that you use ‘maintained’ as the Maintenance section was added in order to ensure that your Work remains useful to the community even when you can no longer maintain and support it yourself.

19.6.4 Derived Works That Are Not Replacements

Several clauses of the LPPL specify means to provide reliability and stability for the user community. They therefore concern themselves with the case that a Derived Work is intended to be used as a (compatible or incompatible) replacement of the original Work. If this is not the case (e.g., if a few lines of code are reused for a completely different task), then clauses 6b and 6d shall not apply.

19.6.5 Important Recommendations

19.6.5.1 Defining What Constitutes the Work : The LPPL requires that distributions of the Work contain all the files of the Work. It is therefore important that

you provide a way for the licensee to determine which files constitute the Work. This could, for example, be achieved by explicitly listing all the files of the Work near the copyright notice of each file or by using a line such as:

```
%% This work consists of all files listed in manifest.txt.
```

in that place. In the absence of an unequivocal list it might be impossible for the licensee to determine what is considered by you to comprise the Work and, in such a case, the licensee would be entitled to make reasonable conjectures as to which files comprise the Work.

20 Files distributed without annotations

20.1 The `prepost.xml` lookup file

This is an example file for customisation to your own style of working. For documentation, see [section 5.2 on page 49](#).

20.2 The `representation.xml` file

This file is used to reproduce the list of typographic representations and the typeface[s] in use. The `<row>` element is tested in `figtab2latex.xml` to compare each named element in the first cell with your *ClassPack* document and the row gets included if there is at least one instance of it in the document (see the fourth item in the list in [section 13 on page 343](#)).

20.3 The `languages.xml` lookup file

This file will be used to look up language details for the `babel` and `polyglossia` packages.

Order is significant in this file. When there are multiple entries for a single `@Lang` attribute, the *last* one is used by *ClassPack*.

20.4 The `readme.xml` template file

This is the boilerplate used to generate the `README.md` file described in [section 11.5.3 on page 261](#).

Change History

v0.1	General: First draft: 1) Construction of .dtx file working; 2) Tested with personal class. 1	Secondary output files possible; reversed usage of role attribute on keywords;. 1
v0.2	General: Used UCC thesis class as test: Devised method for creating bulk options. 1	v0.76 General: Modified documentation: Started working on Makefile. . . 1
v0.3	General: More work with UCC thesis class: Added links to bibliographic styles. 1	v0.77 General: Removed unwanted definitions: classorpackage. . . 1
v0.4	General: Added options: Mostly for controlling appearance. . . . 1	v0.78 General: Changed default code color: MacroFont now DarkBlue. 1
v0.5	General: Bugfixes: Cleaned up XSLT code. 1	v0.79 General: Editorial update: Small corrections. 1
v0.6	General: Changes to date calculations: More use of functions. 1	v0.80 General: Moved doc commands from XSLT2: ToC settings and revmarg. 1
v0.7	General: Testing for self-replication: Danger, Will Robinson. 1	v0.81 General: Added more logo commands: XeTeX, XeLaTeX. . . 1
v0.71	General: First time this was used to document itself: The title element and subtitle element are now subsumed beneath the generated title in the output.. . . 1	v0.82 General: Fixed hline bug: Should have been renewcommand not def. 1
v0.72	General: Wrote internal documentation: Created the classpack.xml template as an example.. . . . 1	v0.9 General: Removed fixltx2e: The code from the fixltx2e package has now been merged into the kernel, so the preload of this package is no longer needed.. . . 1
v0.73	General: Added readme.xml and db2plaintext.xsl: This implements dynamic README generation.. . . . 1	v0.905 General: Enhanced: Added support for Xe ₂ LaTeX. 1
v0.74	General: Added experimental autopackage: This implements automated package inclusion based on the markup used by the author.. . . . 1	v0.91 General: Regenerated: Recreated package with new classpack code to create zip file to the CTAN standard. 1
v0.75	General: Added secondary files:	v0.915 General: Merged: Merged default packages with author-requested ones.. . . . 1
		v0.92 General: Major changes to the way the .dtx code is created: 1) Moved the specification of the processor from an attribute on the root element to the part

start-tag for documentation; 2) Switched from the build file to a Makefile (still incomplete).; 3) Enabled the use of $\text{X}\text{\LaTeX}$ for the documentation in the .dtx file; 4) Largely rewrote the code for auto-detecting and adding packages, combining the auto-detect packages with those specified by the author. It now includes: a) negatable crosslinks specified in the remap on the constructorsynopsis start-tag (eg specifying the use of $\text{X}\text{\LaTeX}$ automatically invokes fontspec and defeats the autoloading of the inputenc and fontenc packages), b) addition of the use of the arch and condition attributes on the bibliography start-tag to hold the $\text{B}\text{\LaTeX}$ package (biblatex) and the $\text{B}\text{\LaTeX}$ engine (biber), c) changes from the text-mode README to a MarkdownREADME.md to accommodate TUG's submission process; 5) The layout of the zip files has been significantly changed to accord with the submission requirements of CTAN; 6) A Bibliography of the documentation is now extracted to disk directly in the XSLT process, rather than via the VerbatimOut environment in the .dtx file; 7) The RFC2119 Note now automatically includes the relevant $\text{B}\text{\LaTeX}$ entry in the .bib file (done in the Makefile). 1

v1.0

General: Tidied up the processing and greatly extended documentation: 1) Rewrote almost all the description of how to create the basic XML file; 2) The RFC2119 Note now automatically includes the relevant $\text{B}\text{\LaTeX}$ entry in the .bib file (done in the Makefile). . 1

v1.01

General: Edited out duplications in documentation: 1) Tidied explanations and documented the markup in more detail; 2) Now using $\text{X}\text{\LaTeX}$ and biber. . . 1

v1.02

General: Moved specification for babel to pre-options: 1) The babel package is now pre-specified with the PassOptionsToPackage command, to avoid conflicts with options later; 2) Now using $\text{X}\text{\LaTeX}$ and biber. 1

v1.025

General: README: Title and identity moved to a template and deleted from readme.xml. . 1

v1.03

General: Bundle identifiers: 1) Added a line to the Makefile to pick out the date of the most-recently-updated file and put it in a file called VERSION (suggestion from Petra Rbe-Pugliese at CTAN based on their docs).; 2) Regularised the identity of the version date from a global variable date, rather than working it out each time.. 1

v1.04

General: Reusable code blocks: 1) Added an attribute reuse to the annotation element for use in Appendixes which generate additional \LaTeX class or package files, which points at an xml:id attribute on an existing annotation element already used elsewhere so that the same documentation text can be included. Also added an omit attribute to the seglistitem elements in the constraintdef element for class packages which specifies that the relevant package is not required for the additional class or package being generated.; 2) Changed the documentation font to Noto.. . . 1

v1.05	General: Maintenance: 1) Updated documentation; 2) Tested additional outputs.	1
v1.06	General: Checked operation of StopEventually	26
	Maintenance release: Rearranged output so that change log and index always get printed.	1
v1.07	General: Minor adaptations to quoted chunks of code: Updated to use \LaTeX	1
v1.08	General: Changed reference to ‘last’ variant to ‘second’ now that the section on mechanically-constructed commands has been removed.	46
	Moved RFC2119 warning, and did some minor rewording: 1) Removed RFC2119 warning text from db2dtx.xml to rfc2119.xml, creating a section to hold it and the bibliography. Rewrote the templates for bibliolist, biblientry, bibliography, and section/title to accommodate this.; 2) Ongoing updates and explanation.	1
	Replaced ‘three’ with ‘two’ uses now that section 2.7 has been removed.	39
v1.09	General: Skipped the application of headers to appendix files: 1) Files written from the appendix element no longer get the LaTeX headers prepended; 2) The sentinel value for comments in scripts now reflects the double hash; 3) Added the remark element to db2md.xml (with plain para); 4) Ongoing updates and explanation.	1
v1.10	General: bold to be plain bold, not bold italic; 3) Changed blockquotes to blue in the PDF, added recognition of xlink:href to give source ack as URI.	1
v1.11	General: 1) Fixed numerous bugs in handling of listings; 2) Moved DTD annotations to doc as chapter while logic for appendixes not working properly; 3) Finished PE chunking of db2dtx.xml and doctexbook-master.dtd; 4) Finished writing the chunk script.	1
v1.12	General: 1) Started tidying up sentinel and fence; 2) Involves rationalising the use of chapter and appendix within part; 3) parameterised a makesentinel, maketermsentinel, and makefence in sect1 but they don’t get called because of [2] above.	1
v1.13	General: 1) Created commentchar and termcommentchar functions; 2) Untested invocation in sect1 template. . .	1
v1.14	General: 1) Updated the Markdown driver for the README to produce code that displays correctly in the Markdown Viewer 3.6 extension for Chrome and conforms to the CTAN rules.; 2) Added details of ClassPack to the MD file header; 3) Some reorganisation of topics; 4) More annotation-level documentation on the XSLT code.	1
v1.15	General: 1) Started massive documentation effort on XSLT; 2) Order of templates changed in XSLT.	1
v1.16	General: 1) Documentation ongoing; 2) Added detection of options on url and xcolor; 3) Changed order of packages in prepost.xml to prevent option clashes.	1

- v1.17
General: 1) Documentation nearly finished; 2) Added remaining XSLT files and scripts; 3) Recoded the distinction between appendixes in the code part and chapters in the files part.. 1
- v1.18
General: 1) Replaced conformance with YYYY-MM-DD on all date elements in the revision history. This means a complete regression run after editing all source masters to use this change; 2) Fixed bug in makechapapp where it failed to add the filetype before substringing the full name of the document; 3) Split project into classpack and classpack-dev so as not to overload the regular user with stuff they don't need. 1
Fixed bug where it failed to add the filetype before substringing the full name of the document 249
This changed from the conformance attribute to the YYYY-MM-DD attribute.
Updated all master documents. 26
- v1.19
General: 1) Added hypdoc package as default for documentation in prepost.xml to overcome makeindex bug using hyperref (thanks to Ulrike Fischer on c.t.t); 2) Replaced the tabular used for the forced inclusion of the first three lines of the class or package in (in order to maintain the line-count) with three plain lines because something in hyperref was messing with DocTeX in a way that hypdoc doesn't; 3) Rewrote documentation about how makepackages constructs the list of candidate element types. 1
- v1.20
General: Major edit of user documentation to remove duplications.. 1
- v1.21
General: Re-ordered and documented font selection in prepost.xml. 1
- v1.22
General: Rearranged documentation order, moved regular classpack.sty code back into the document in preparation for merging the dev code and going back to a single package. Corrected errors in autopackage and rewrote the XPath for detecting matches from prepost.xsl.. . . . 1
- v1.23
General: Finished repairs to the extracting routines for code, added a bypass to allow for deferred comments at the top of shell scripts to avoid trampling on the hashbang. Renamed the development package back to classpack and re-ran all tests.. . 1
- v1.24
General: 1) Rationalised the transformation of autopackage packages from prepost.xml format to document format; 2) Split the XPath that finds them into six segments so that the finds can be categorised properly; 3) Rewrote the transformation from the packages tree into the \LaTeX serialisation (with new routines for preliminary RequirePackage and for PassOptionsToPackage), using the rationalised attributes on the new packages tree.. . . . 1
- v1.25
General: "draft" would be picked up (all the other tests started too far down the tree).. 1
- v1.26
General: Regression release for revised db2dtx.xsl 3.1. 1
- v1.27
General: 1) Rewrite of the way in which packages are detected and handled for the

AutoPackage mechanism; 2) Removed (commented out) the code for the widening of the LH margin in mid-document to accommodate long macro names. For the moment it uses fixed margins all the way through. The adjustment feature may return if problems with the formatting of other page elements (particularly running headers and footers) can be resolved.; 3) Ongoing work to find out why the dox package is interpreting endmacrocode when it occurs in mid-code and not at the start of the line.; 4) Minor corrections to formatting. 1

v1.28

General: 1) Full regression test on all maintained packages. AutoPackage methoology being

documented separately; 2) The code for the widening of the LH margin in mid-document (removed in 1.27) remains in abeyance until some of the discrepancies with fancyhdr headings are resolved; 3) The problem with the dox package interpreting endmacrocode when it occurs in mid-code and mid-line has been temporarily circumvented by splitting the relevant xsl:text to output the command in two pieces; 4) Additional discrepancy discovered of DocTeX parsing an ifcase command mentioned in a comment behind a double percent signs in the .dtx file. The package text has been reworded to avoid this until we find out what's happening. 1

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

`\#` 96,
609, 839, 965, 981, 4306, 6035, 6036
`\$` 95, 840, 965, 981, 7616,
7732, 7786, 7793, 7839, 7846, 7853
`\%` 840, 958, 981, 6023, 6024
`\&` 75, 106, 840, 844, 965, 966,
981, 4300, 4306, 4353, 4471, 6031
`\@pnumwidth` 29
`\@tocrmarg` 30
`\{` 367, 822, 823, 824, 848,
959, 972, 978, 984, 990, 996,
1002, 1008, 1014, 1020, 1026,
2751, 6959, 6969, 6986, 6991, 7034
`\}` 367, 822, 823, 824, 849,
959, 972, 978, 984, 990, 996,
1002, 1008, 1014, 1020, 1026,
2751, 6961, 6971, 6986, 6991, 7034
`\^` 847, 981, 7616, 7732
`_` 847, 966, 981, 8260
`\|` 848, 931, 981, 2457, 7616, 7732
`\~` 849, 931, 981
"1.0" (attribute value) 19
"12/14" (attribute value) 234
"2014 EPDIC" (attribute value) 75
"7/8" (attribute value) 331

Numbers

"1" (attribute value) 64, 343

`_` 50,
54, 58, 62, 98, 99, 100, 112, 115,
931, 1120, 1309, 1330, 3230,
5187, 6023, 6024, 6031, 6035,
6036, 7044, 8964, 8965, 8966, 8967

A

`<abstract>` (element) 25
`<abstract>` (element) 26, 170
`abstract` (template) 1204
`<acronym>` (element) 65
`addindent` (template) 273, 7957
`<address>` (element) 74, 75
`address` (template) 1209
"affil" (attribute value) 333
`<affiliation>` (element) 24, 73
"after" (attribute value) 44, 257

`@align` (attribute) 54, 64, 336, 340
"all" (attribute value) 22
"alpha" (attribute value) 22
`<anchor>` (element) 32, 364
`<annotation>` (element) 25, 217
`<annotation/>` (empty element) ... 29
`<annotation>` (element)
..... 16, 27–30, 37, 59, 149,
153, 154, 156, 201, 211, 240, 389
`@annotations` (attribute) 23, 61
`<annotations>` (element) 23
`@annotations` (attribute)
..... 61, 65, 109, 115, 156, 182, 234
"apa" (attribute value) 71
"apacite" (attribute value) ... 71, 316
' (general entity) 299
`appdir` (variable) 93
`<appendix>` (element) 21, 30, 38
`\appendix` 36
`appendix` (template) 249
`<apply-templates>` (element) 376
`@arch` (attribute) 22, 30, 40, 61, 71
`<arch>` (element) 22
`@arch` (attribute)
29, 40, 51, 71, 96, 103, 108, 149,
151, 156, 163, 216, 234, 263, 351
`armour` (parameter) 30
`armour` (variable) 307
"article" (attribute value) 72
`article@article` (class) 45, 57, 66
`<artpagenums>` (element) 74
"atbegindoc" (attribute value) . 34, 45
"atendpreamble" (attribute value) ..
..... 34, 45
`attribmatch` (variable) 5408
"1" 64, 343
"1.0" 19
"12/14" 234
"2014 EPDIC" 75
"7/8" 331
attribute values:="" 23
"affil" 333
"after" 44, 257
"all" 22
"alpha" 22
"apa" 71
"apacite" 71, 316

"article"	72	"LaTeX"	67
"atbegindoc"	34, 45	"LaTeX2e"	22
"atendpreamble"	34, 45	"LaTeXe"	61
"author"	65	"length"	69, 175, 201
"awk"	30	"lpp1"	22
"b"	68, 339	"lualatex"	218
"bash"	22, 66, 69	"m"	68, 339
"before"	44, 257	"maintainer"	25
"beta"	22	"manifest"	46
"biber"	71	"math"	51
"biblatex"	71, 103, 109	"mono"	51
"BiBTeX"	69	"monospace"	68
"bibtex"	71	"n"	68
"book"	72	"new"	283
"bugs"	363	"newpage"	343
"bx"	68	"news"	200
"c"	68	"noat"	68
"ch"	65	"nodesc"	65, 201
"chapter"	72	"nolit"	267
"char"	54, 340	"nosc"	52
"class"	22	"off"	109, 226
"cls"	22, 51, 53, 109, 239	"only"	41, 242
"clspackages"	37, 39, 41, 52	"options"	37, 44
"code"	27	"other"	74
"code"	21, 30, 59–61, 129, 133, 135, 146, 156, 174, 250, 254	"overhang"	334
"color"	69	"p"	65
"compact"	39, 60	"package"	22
"counter"	69, 175, 201	"page"	70
"cp"	30	"page"	57
"data"	132, 133, 211, 333, 341	"paper"	72
"doc"	26	"parameter"	67
"doc"	21, 26, 36, 51, 53, 55, 56, 58, 61, 70, 103, 109, 129, 216, 239	"paren"	65
"docpackages"	39, 113, 226	"pdflatex"	58
"document"	30	"pkg"	239
"draft"	22	"pl"	22
"dtd"	30	"postpackage"	50, 55
"field"	65, 69	"prepackage"	50, 55, 241
"file"	156	"proceedings"	72
"files"	30	"py"	22
"files"	21, 30, 70, 129, 133, 135	"r"	68
"final"	22	"rc"	22
"footer"	330	"roman"	68
"frag"	62, 156	"ruled"	343
"framed"	61, 234, 351	"sans"	51, 68
"full"	65	"sc"	68
"hand"	51	"script"	22
"header"	64, 343	"serif"	51
"ignore"	61	"short"	57
"it"	68	"shortauthor"	65
"journal"	72	"shorttitle"	65
"k"	68	"sl"	68
"l"	68	"sponsor"	25
		"strong"	177
		"sty"	22, 51, 53, 109

- "stypackages" 37, 39, 41, 52
- "switch" 175
- "symbol" 51
- "template" 201
- "test" 343
- "TeX" 67
- "text" 57, 65
- "thead" 336
- "title" 65, 66
- "TUG'95" 75
- "up" 68
- "uri" 74
- "variable" 201
- "widest" 339
- "xelatex" 55, 58, 109, 216, 218
- "XML" 66, 69
- "xsl" 30
- "XSLT" 66, 69
- attributes:
 - fontface [58](#)
 - label [19](#)
 - language [40](#)
 - linkend [17](#)
 - omit [66](#)
 - role [30](#)
 - units [11](#)
 - version [38](#)
 - YYYY-MM-DD [7](#)
- @align 54, 64, 336, 340
- @annotations 23, 61
- @annotations
 - 61, 65, 109, 115, 156, 182, 234
- @arch 22, 30, 40, 61, 71
- @arch 29, 40, 51, 71, 96, 103, 108, 149,
 - 151, 156, 163, 216, 234, 263, 351
- @audience 22
- @audience 29, 149
- @begin 16, 65
- @char 63
- @char 339
- @class 68, 74, 87, 192
- @colwidth 63
- @colwidth 336, 339
- @condition 22, 61, 63, 71
- @condition . 41, 50–53, 65, 101, 102,
 - 109, 156, 174, 178, 201, 216,
 - 220, 242, 336, 343, 352, 354, 360
- @conformance 22, 63
- @conformance 22, 23,
 - 36, 51, 55, 62, 68, 96, 101, 109,
 - 111, 156, 174, 178, 216, 218, 340
- @constructorsynopsis 217
- @coverart 33
- @depth 352
- @end 16, 65
- @endinglinenumber 62
- @endinglinenumber 16, 61, 62, 156
- @endterm 70
- @fileref 53, 63, 217, 352
- @filetype 263
- @floatstyle 63
- @fontalign 17
- @fontcolor 68
- @fontcolor 17, 69
- @fontencoding 68
- @fontencoding 17
- @fontface 68
- @fontface 17
- @fontfamily 68
- @fontfamily 17
- @fontseries 68
- @fontseries 17
- @fontshape 68
- @fontshape 17
- @fontsize 68
- @fontsize 17, 68
- @funcparams 54
- @href 146
- @label 71
- @label 16, 71, 86, 316
- @Lang 401
- @language 61
- @language ... 16, 60, 66, 67, 69, 156, 180
- @linenumbering 156, 234
- @linkend 16, 34, 37, 44, 54, 57,
 - 60, 65, 66, 69, 70, 103, 173, 204, 219
- @match 152
- @omit 17
- @os 22
- @parameter 54
- @relation 73
- @remap 23, 61
- @remap 50, 51,
 - 54, 65, 108, 109, 143, 156, 175,
 - 182, 234, 307, 314, 334, 343, 354
- @reuse 16, 29, 153, 154
- @revision 22, 26
- @revision 16, 19, 33, 36, 51, 65, 76
- @role 16, 17,
 - 25, 28, 29, 34, 41, 51, 57, 64, 65,
 - 69, 86, 109, 117, 132, 149, 150,
 - 153, 156, 167, 175, 177, 178,
 - 180, 187, 192, 201, 240, 257,
 - 304, 309, 312, 315, 330, 331, 343
- @rowsep 64, 343
- @security 19, 23
- @security 19, 33, 76
- @show 283

@spacing 39, 60
 @spanname 64
 @startinglinenumber 62
 @startinglinenumber . 61, 62, 156, 236
 @status 22
 @style 343
 @tgroupstyle 334
 @token 159
 @type 200
 @units 16, 65
 @userlevel 22, 30
 @userlevel 30, 50, 67, 109, 190, 226, 263
 @varname 67
 @vendor 23
 @vendor 51
 @version 19, 22
 @version .. 16, 19, 33, 36, 41, 65, 76, 109
 @width 63, 352
 @wordsize 49, 61, 64, 68
 @wordsize ... 64, 147, 156, 234, 331, 336
 @xlink:href 30, 62, 71
 @xlink:href 30,
 31, 36, 38, 61, 67, 113, 156, 182,
 184, 200, 230, 249, 283, 307, 333
 @xlink:role 23, 41
 @xlink:role 41, 112, 242
 @xlink:show 156
 @xml:base 23
 @xml:id 21, 30
 @xml:id 16,
 18, 21, 23, 24, 29, 30, 34, 37, 39,
 40, 44–46, 54, 56–58, 60, 65, 66,
 69, 71, 76, 94, 128, 129, 143, 146,
 154, 156, 168, 170, 183, 234, 330
 @xml:lang 49, 66, 67
 @xref 159
 @xreflabel 70, 71
 @xreflabel 16, 23, 29, 66, 70,
 71, 80, 86, 149, 156, 200, 201, 216
 @xrefstyle 70
 @xrefstyle 57, 65, 66
 @YYYY-MM-DD 16, 26, 35, 73–76
 @YYYY-MM-DD-from 75
 @YYYY-MM-DD-to 75
 attvalmatch (variable) [5438](#)
 attvalptrmatch (variable) [5482](#)
 @audience (attribute) 22
 <audience> (element) 185
 @audience (attribute) 29, 149
 <author> (element) 24
 "author" (attribute value) 65
 <author> (element) 24, 25, 73
 <authorgroup> (element) ... 25, 35, 73
 avoidverb (template) [175](#), [179](#), [201](#), [7601](#)

"awk" (attribute value) 30

B

"b" (attribute value) 68, 339
 "bash" (attribute value) 22, 66, 69
 "before" (attribute value) 44, 257
 @begin (attribute) 16, 65
 BerryKarl 17, 68
 "beta" (attribute value) 22
 "biber" (attribute value) 71
 "biblatex" (attribute value)
 71, 103, 109
 biblioallcite (template) [310](#)
 biblioauthed (template) [524](#)
 biblioauthgroup (template) [497](#)
 biblioauthsolo (template) [542](#)
 biblioauthterm (template) [611](#)
 bibliocorpauth (template) [618](#)
 <bibliocoverage> (element) 327
 bibliodata (template) [381](#)
 <biblioentry> (element)
 16, 37, 57, 60, 65, 70, 71, 300
 biblioentry (template) [413](#)
 bibliofile (template) [327](#)
 <bibliography> (element) 71
 <bibliography> (element)
 16, 57, 59, 70, 71, 103, 113
 bibliography (template) [298](#)
 <biblioid> (element) 16, 74
 <biblioitem> (element) 59
 <bibliolist> (element) 306
 bibliolist (template) [296](#)
 <bibliomisc> (element) 310
 bibliomisc (template) [455](#)
 biblionewpage (template) [321](#)
 biblioorgs (template) [655](#)
 biblioprintbiblatex (template) .. [338](#)
 biblioprintbibtex (template) ... [360](#)
 <biblioref> (element) .. 60, 65, 70, 299
 <biblioref> (element)
 16, 57, 66, 70, 71, 299, 306
 <biblioset> (element) . 72–74, 310, 315
 bibliosets (template) [460](#)
 biblioshorttitles (template) [794](#)
 bibliosources (template) [799](#)
 bibliosubtitles (template) [720](#)
 bibliotitles (template) [671](#)
 \BIBTeX [86](#)
 "BiBTeX" (attribute value) 69
 "bibtex" (attribute value) 71
 <blockquote> (element) 59, 65
 <blockquote> (element)
 16, 57, 59, 60, 65, 306
 "book" (attribute value) 72

<book> (element) . 20, 21, 24, 26, 33,
36, 40, 41, 49, 76, 95, 112, 115, 242
<book> (start-tag) 20, 24
breakline (template) 94, [7978](#)
<bridgehead> (element) 60
<bridgehead> (element) 59, 80
"bugs" (attribute value) 363
"bx" (attribute value) 68

C

"c" (attribute value) 68
casestyle (template) [6339](#)
"ch" (attribute value) 65
"chapter" (attribute value) 72
<chapter> (element) 21, 22, 26,
27, 30, 31, 37, 38, 44, 58, 70, 94, 132
chapter (template) 249
@char (attribute) 63
"char" (attribute value) 54, 340
@char (attribute) 339
checkpackages (template) . . . 256, [7175](#)
chunk (function) [1](#)
chunkawk (function) [3](#)
chunkawkscript-comment (com-
ment) [1](#)
chunkscrip-comment (comment) . . . [2](#)
citeauthors (template) [117](#)
citedetail (template) [29](#)
citelink (template) [273](#)
citeother (template) [163](#)
citeshort (template) [63](#)
citesurnames (template) [153](#)
<citetitle> (element) 66
<citetitle> (element) 306
citetitle (template) [34](#)
citeunits (template) [240](#)
"class" (attribute value) 22
@class (attribute) 68, 74, 87, 192
class@class (class) 8
classes:
 euroflag 174
classes:article@article 45, 57, 66
classes:class@class 8
classes:ltxdoc@ltxdoc 45, 48
<classname> (element) 66
<classname> (element) 17, 57, 178
clean (file) [4](#)
"cls" (attribute value)
 22, 51, 53, 109, 239
"clspackages" (attribute value) . . .
 37, 39, 41, 52
<cmdsynopsis> (element)
 17, 55, 113, 121, 276
"code" (attribute value) 27

<code> (element) 66
"code" (attribute value)
 21, 30, 59–61, 129,
 133, 135, 146, 156, 174, 250, 254
"color" (attribute value) 69
colprefix (template) 334, [356](#)
colsettings (template) 334, [489](#)
<colspec> (element)
 54, 63, 64, 332, 334, 340
colsuffix (template) 334, [571](#)
columnsep (length) 80, [109](#)
@colwidth (attribute) 63
<colwidth> (element) 64
@colwidth (attribute) 336, 339
<command> (element) 66
<command> (element)
 17, 55, 57, 69, 117, 178, 241
commentchars (variable) 95, 263
comments:
 chunkawkscript-comment [1](#)
 chunkscrip-comment [2](#)
 cpdtdoc-comment [1](#)
 db2bibtexscript-comment [1](#)
 db2dtxscript-comment [1](#)
 db2mdscript-comment [1](#)
 decommentscript-comment [1](#)
 figtabscript-comment [1](#)
 usemakefile-comment [1035](#)
"compact" (attribute value) 39, 60
Comprehensive T_EX Archive Network
 see CTAN
compspace (template) [8038](#)
@condition (attribute) 22, 61, 63, 71
<condition> (element) 152
@condition (attribute)
 41, 50–53, 65, 101, 102,
 109, 156, 174, 178, 201, 216,
 220, 242, 336, 343, 352, 354, 360
<confdates> (element) 16, 75
confdates (template) [1065](#)
<confgroup> (element) 74
confgroup (template) [994](#)
<confnum> (element) 75
conforgaddress (template) [1055](#)
@conformance (attribute) 22, 63
<conformance> (element) 58
@conformance (attribute) 22, 23,
 36, 51, 55, 62, 68, 96, 101, 109,
 111, 156, 174, 178, 216, 218, 340
<confsponsor> (element) 75
confsponsor (template) [1045](#)
<conftitle> (element) 73
conftitle (template) [999](#)
<constraintdef> (element) 40

`<constraintdef>` (element)
 34, 37, 39, 40, 44–46, 52,
 55, 113, 153, 226, 239, 241, 257, 333
`<constructorsynopsis>` (element) .
 52, 54, 102, 103, 109, 216
`@constructorsynopsis` (attribute) . 217
`\ConTeXt` 40
`<contrib>` (element) 25
`<copyright>` (element) 25
`copyright` (template) 8064
`<coref>` (element) 54, 66
`<coref>` (element) 54, 84
`corefs` (template) 4725
`"counter"` (attribute value) 69, 175, 201
counters:
 `CPKcoref` 119
 `CPKfnote` 111
 `IndexColumns` 13, 108
 `secnumdepth` 106
 `tocdepth` 107
`<cover>` (element) 24, 39
`coverage` (template) 1232
`@coverart` (attribute) 33
`"cp"` (attribute value) 30
`cpdir` (variable) 93
`cpdtddoc-comment` (comment) 1
`CPKcoref` (counter) 119
`CPKfnote` (counter) 111
`\CPKmenusep` 98
`\CPKpoststrut` 101
`\CPKprestrut` 100
`\CPKrunningecho` 18
`\CPKtabwid` 15, 89, 110, 227
`CPKtabwid` (length) 110
`\CPKvstrut` 96
`CTAN` 1, 11,
 22, 25, 31, 35, 92, 355, 379, 383, 384
`ctan` (file) 4

D

`"data"` (attribute value)
 132, 133, 211, 333, 341
`<date>` (element) 16, 26, 73, 76
`dates` (template) 1081
`db2bibtexscript-comment` (com-
 ment) 1
`db2dtxscript-comment` (comment) .. 1
`db2mdscript-comment` (comment) ... 1
`db5dtd` (entity) 3
`db:abstract` (template) 1076
`db:acknowledgements` (template) . 2046
`db:acronym` (template) 486, 3273
`db:affiliation` (template) 1428
`db:anchor` (template) 404

`db:annotation` (template)
 2353, 2563, 5131
`db:appendix` (template) 1725
`db:author` (template) 1394
`db:biblioref` (template) 10, 694
`db:blockquote` (template) 2234
`db:chapter` (template) 1625
`db:citetitle` (template) 3397
`db:classname` (template) 3460
`db:cmdsynopsis` (template) 1113
`db:code` (template) 3435
`db:command` (template) 582, 3508
`db:constraintdef` (template) ... 2511
`db:declopts` (template) 5155
`db:email` (template) 3613
`db:emphasis` (template) 3597
`db:envvar` (template) 3618
`db:errortext` (template) 3647
`db:exceptionname` (template) 267, 3655
`db:filename` (template) 3663
`db:firstterm` (template) 3672
`db:footnote` (template) 523, 3680
`db:foreignphrase` (template) ... 3685
`db:formalpara` (template) .. 206, 3258
`db:function` (template) 3727
`db:guibutton` (template) 3833
`db:guiicon` (template) 3802
`db:guilabel` (template) 3827
`db:guimenu` (template) 3741
`db:guimenuitem` (template) 3784
`db:guisubmenu` (template) 3766
`db:honorific` (template) 1420
`db:important` (template) 3841
`db:indexterm` (template) 3846
`db:info` (template) 1266
`db:itemizedlist` (template) 2873
`db:keycap` (template) 3856
`db:link` (template) 3883
`db:listitem` (template) 3171
`db:literal` (template) 2225
`db:literallayout` (template) 457, 3935
`db:markup` (template) 4222
`db:methodname` (template) 3981
`db:note` (template) 3992, 4006
`db:olink` (template) 454
`db:option` (template) 4021
`db:orderedlist` (template) 2930
`db:package` (template) 4061
`db:para` (template) 312, 354, 2060
`db:part` (template) 1476
`db:personname` (template) 4097
`db:pi` (template) 5118
`db:preface` (template) 1450
`db:primary` (template) 4136

db:productname (template) [4143](#)
 db:programlisting (template)
 [229](#), [2623](#)
 db:quotes (template) [4154](#)
 db:releaseinfo (template) [1446](#)
 db:remark (template) [2570](#)
 db:replaceable (template) [4167](#)
 db:sect1 (template) [386](#), [1850](#)
 db:sect2 (template) [1919](#)
 db:sect3 (template) [1956](#)
 db:sect4 (template) [1984](#)
 db:section (template) [2012](#)
 db:see (template) [4212](#)
 db:simplelist (template) [3125](#)
 db:superscript (template) [4217](#)
 db:tag (template) [592](#), [4228](#)
 db:task (template) [4518](#)
 db:term (template) [281](#), [3209](#)
 db:title (template) [52](#), [1284](#), [2030](#), [4534](#)
 db:token (template) [4539](#)
 db:type (template) [4545](#)
 db:typecol (template) [4642](#)
 db:uri (template) [513](#), [4650](#)
 db:userinput (template) [2867](#)
 db:variablelist (template) [270](#), [3083](#)
 db:varname (template) [4675](#)
 db:warning (template) [151](#), [2264](#)
 db:xref (template) [528](#), [4802](#)
 <dd> (element) 363
 decommentbbl (function) [3](#)
 decommentscript-comment (com-
 ment) [1](#)
 default (template) [777](#), [5358](#)
 defaultmatch (variable) [5508](#)
 delogify (template) [8271](#)
 dependencies (setting) [4](#)
 dependencymatch (variable) [5513](#)
 @depth (attribute) 352
 \descriptionlabel [119](#)
 dest (parameter) 239
 detex (template) [966](#)
 <dl> (element) 361
 "doc" (attribute value) 26
 "doc" (attribute value)
 21, 26, 36, 51, 53, 55, 56,
 58, 61, 70, 103, 109, 129, 216, 239
 docname (variable) 94
 "docpackages" (attribute value) ...
 39, 113, 226
 "document" (attribute value) 30
 document class **10**
 Document Type Declaration **19**
 Document Type Definition ... see DTD
 dodate (template) [1084](#)

dologo (template) 376, [944](#)
 done (template) [1237](#)
 "draft" (attribute value) 22
 <dt> (element) 361
 DTD . 10, 11, 15, 16, 18, 19, **19**, 20, 28,
 56, 71, 85, 135, 139, 146, 156, 170
 "dtd" (attribute value) 30
 DTDs/Schemas:
 html [5](#)
 dtx (file) [4](#)
 dtxttext (template) [214](#), [8207](#)

E

<edition> (element) 74
 edition (template) [1172](#)
 <editorgroup> (element) 73
 element types:
 seglistitem
 [5574](#), [5598](#), [5657](#), [5701](#), [5747](#), [5770](#)
 xsl:output [18](#)
 xsl:stylesheet [3](#)
 elementmatch (variable) [5387](#)
 elements (variable) [217](#), [221](#), [5539](#)
 <abstract> 25
 <abstract> 26, 170
 <acronym> 65
 <address> 74, 75
 <affiliation> 24, 73
 <anchor> 32, 364
 <annotation> 25, 217
 <annotation> 16, 27–30, 37, 59, 149,
 153, 154, 156, 201, 211, 240, 389
 <annotations> 23
 <appendix> 21, 30, 38
 <apply-templates> 376
 <arch> 22
 <artpagenums> 74
 <audience> 185
 <author> 24
 <author> 24, 25, 73
 <authorgroup> 25, 35, 73
 <bibliocoverage> 327
 <biblioentry>
 16, 37, 57, 60, 65, 70, 71, 300
 <bibliography> 71
 <bibliography>
 16, 57, 59, 70, 71, 103, 113
 <biblioid> 16, 74
 <biblioitem> 59
 <bibliolist> 306
 <bibliomisc> 310
 <biblioref> 60, 65, 70, 299
 <biblioref> . 16, 57, 66, 70, 71, 299, 306
 <biblioset> 72–74, 310, 315

- `<blockquote>` 59, 65
- `<blockquote>` ... 16, 57, 59, 60, 65, 306
- `<book>` 20, 21, 24, 26, 33,
36, 40, 41, 49, 76, 95, 112, 115, 242
- `<bridgehead>` 60
- `<bridgehead>` 59, 80
- `<chapter>` 21, 22, 26,
27, 30, 31, 37, 38, 44, 58, 70, 94, 132
- `<citetitle>` 66
- `<citetitle>` 306
- `<classname>` 66
- `<classname>` 17, 57, 178
- `<cmdsynopsis>` ... 17, 55, 113, 121, 276
- `<code>` 66
- `<colspec>` 54, 63, 64, 332, 334, 340
- `<colwidth>` 64
- `<command>` 66
- `<command>` .. 17, 55, 57, 69, 117, 178, 241
- `<condition>` 152
- `<confdates>` 16, 75
- `<confgroup>` 74
- `<confnum>` 75
- `<conformance>` 58
- `<confsponsor>` 75
- `<conftitle>` 73
- `<constraintdef>` 40
- `<constraintdef>`
..... 34, 37, 39, 40, 44–46, 52,
55, 113, 153, 226, 239, 241, 257, 333
- `<constructorsynopsis>`
..... 52, 54, 102, 103, 109, 216
- `<contrib>` 25
- `<copyright>` 25
- `<coref>` 54, 66
- `<coref>` 54, 84
- `<cover>` 24, 39
- `<date>` 16, 26, 73, 76
- `<dd>` 363
- `<dl>` 361
- `<dt>` 361
- `<edition>` 74
- `<editorgroup>` 73
- `<email>` 102
- `<emphasis>` 66
- `<entry>` 64, 340, 343, 345
- `<envar>` 66
- `<envar>` 17, 57
- `<exceptionname>` 66
- `<exceptionname>` 121, 306, 381
- `<figure>` 62
- `<figure>` 57, 59, 67
- `<filename>` 66
- `<files>` 384
- `<firstname>` 67, 73, 189
- `<firstterm>` 66
- `<footnote>` 66
- `<foreignphrase>` 66
- `<form>` 85
- `<formalpara>` 359
- `<function>` 66
- `<function>` 17, 57
- `<gui*>` 181
- `<guibutton>` 66
- `<guilabel>` 66
- `<guimenu>` 66
- `<guimenuitem>` 66
- `<guisubmenu>` 66
- `<holder>` 25
- `<html:form>` 16, 85
- `<ID>` . 57, 70, 94, 249, 257, 300, 363, 366
- `<IDREF>` 16, 70, 257, 300
- `<IDREFS>` 16, 29, 57, 300
- `<imagedata>` 53, 63, 67, 217, 351
- `<imageobject>` 63, 67, 354
- `<indexterm>` 190, 191
- `<info>` 24, 27, 33, 97, 332
- `<informaltable>` 64, 70
- `<informaltable>` 57, 59, 64, 70
- `<inlineequation>` 66
- `<inlinemediaobject>` 67
- `<issuenum>` 74
- `<itemizedlist>` 60, 67, 69
- `<itemizedlist>` . 48, 56, 57, 59, 60, 163
- `<keycap>` 67
- `<link>` 67, 69
- `<link>` 69, 102
- `<linkend>` 85
- `<listitem>` 69, 70
- `<listitem>` 54, 56, 57, 60, 70, 169
- `<literal>` 67
- `<literal>` 17
- `<literallayout>` 67
- `<literallayout>` 17, 32
- `<markup>` 67
- `<mediaobject>` 63
- `<member>` 46
- `<methodname>` 67
- `<methodparam>` 53, 54, 102
- `<methodsynopsis>` 211
- `<modifier>` 53, 102
- `<none>` 185
- `<note>` 64, 70
- `<note>` 57, 59
- `<olink>` 31, 360
- `<option>` 67
- `<option>` 17, 57
- `<orderedlist>` 60, 67, 70
- `<orderedlist>` ... 48, 56, 57, 59, 60, 70

- `<orderedlist/listitem>` 70
- `<org>` 74
- `<orgdiv>` 73, 74
- `<orgname>` 73, 74
- `<otherclass>` 74
- `<othername>` 73
- `<package>` 67
- `<package>` 57
- `<packages>` 109
- `<pagenums>` 74
- `<para>` 59
- `<para>` 27, 32, 36, 50, 52, 56, 57, 59, 60, 65
- `<parameter>` 67
- `<parameter>` ... 17, 53, 54, 57, 102, 201
- `<part>` 70
- `<part>` 21, 27, 30, 53, 55, 56, 58, 70, 96, 132, 218
- `<personname>` 24, 73
- `<phrase>` 67
- `<preface>` 58
- `<prepackage>` 216
- `<procedure>` 70
- `<procedure>` 49–52, 56, 57, 65, 109
- `<procedure/step>` 70
- `<productname>` 67
- `<productnumber>` 325
- `<programlisting>` 70
- `<programlisting>` 16, 17, 27, 30, 31, 38, 49, 57, 59–61, 70, 190, 198, 217, 230, 273, 279, 283, 288, 385
- `<publisher>` 74
- `<publishername>` 74
- `<quote>` 67
- `<quote>` 16
- `<refsection>` 49
- `<releaseinfo>` 25
- `<releaseinfo>` 35, 321
- `<remark>` 65
- `<remark>` 16, 26, 59, 154
- `<replaceable>` 68
- `<replaceable>` 267
- `<repleaceable>` 270
- `<revdescription>` 36
- `<revhistory>` 19, 22, 23, 26
- `<revhistory>` 23, 76, 115
- `<revision>` 19, 22, 76
- `<role>` 25, 27, 44, 69, 242
- `<row>` 64, 343, 401
- `<screen>` 17
- `<sect1>` 60
- `<sect1>` 26, 27, 30, 32, 44, 45, 58–60, 139, 250, 256, 363
- `<sect2>` 60
- `<sect2>` 26, 58–60, 140
- `<sect3>` 60
- `<sect3>` 58–60
- `<section>` 58, 141, 306
- `<seg>` 33, 40, 41, 52, 109, 239
- `<seglistitem>` 17, 33, 40, 41, 51, 52, 108, 109, 221, 227, 239
- `<segmentedlist>` 40, 45, 50
- `<segtitle>` 41, 45
- `<sidebar>` 64, 70
- `<sidebar>` 57, 59
- `<simplelist>` 46
- `<status>` 22
- `<step>` 70
- `<step>` 50–53, 55, 57, 65, 103, 109, 216, 221
- `<subtitle>` 59
- `<subtitle>` 59, 73, 126
- `<surname>` 67, 73, 189
- `<systemitem>` 68
- `<table>` 63
- `<table>` 57, 59
- `<tag>` 56, 68
- `<tag>` 87
- `<targetptr>` 32
- `<task>` 64
- `<task>` 59, 65
- `<tbody>` 63, 332, 333, 343
- `<term>` 60, 169, 175, 361
- `<text()>` 376
- `<textobject>` 330, 354
- `<tfoot>` 63, 332, 345
- `<tgroup>` 43, 64, 103, 331, 332, 334
- `<thead>` 332, 341, 345
- `<title>` 24, 59
- `<title>` 24, 26, 35, 58, 59, 64, 71, 73, 126, 170, 300, 330, 332
- `<titleabbrev>` 73
- `<token>` 190
- `<type>` 68, 69
- `<type>` 17, 32, 69, 200
- `<uri>` 69
- `<uri>` 102
- `<userinput>` 160, 234
- `<userlevel>` 22
- `<variablelist>` 60
- `<variablelist>` 56, 57, 59, 60, 168, 361, 363
- `<varlistentry>` 69
- `<varlistentry>` 54, 57, 60, 69, 169, 219
- `<varname>` 69
- `<varname>` 17, 57, 86
- `<volumenum>` 74
- `<warning>` 64, 70

- `<warning>` 57, 59
 - `<wordasword>` 69
 - `<xml:id>` 66
 - `<xref>` 69
 - `<xref>` 16, 53, 54, 57, 204, 219
 - `<xreflabel>` 27
 - `<xsl:apply-templates>` 159, 267
 - `<xsl:include>` 299
 - `<xsl:otherwise>` 132, 142
 - `<xsl:result-document>` 92
 - `<xsl:stylesheet>` 91
 - `<xsl:value-of>` 159
 - `<year>` 16, 25
 - `<email>` (element) 102
 - `<emphasis>` (element) 66
 - EMPTY (local name) 16, 29, 61, 75
 - `<annotation/>` 29
 - `<programlisting/>` 31
 - `<tag/>` 8
 - `@end` (attribute) 16, 65
 - `end` (template) [1034](#), [9144](#)
 - `\endhead` [604](#), [780](#)
 - `@endinglinenumber` (attribute) 62
 - `@endinglinenumber` (attribute)
..... 16, 61, 62, 156
 - `endtag` (template) [7556](#)
 - `@endterm` (attribute) 70
 - entities:
 - `db5dtd` [3](#)
 - HTML [118](#)
 - invocation [186](#)
 - LaTeX [74](#)
 - OUTPUT [71](#)
 - `<entry>` (element) ... 64, 340, 343, 345
 - `entrybody` (template) [902](#)
 - `entryheadfoot` (template) [786](#)
 - `<envar>` (element) 66
 - `<envar>` (element) 17, 57
 - `eurolag` (class) 174
 - `<exceptionname>` (element) 66
 - `<exceptionname>` (element)
..... 121, 306, 381
 - `\ExecuteOptions` [1905](#)
 - `extinc` (template) [2809](#)
- F**
- `fancyhdr` (package) [5](#)
 - `"field"` (attribute value) 65, 69
 - `figtabscript-comment` (comment) .. [1](#)
 - `<figure>` (element) 62
 - `<figure>` (element) 57, 59, 67
 - `figure` (template) [1014](#)
 - `"file"` (attribute value) 156
 - `<filename>` (element) 66
 - `@fileref` (attribute) ... 53, 63, 217, 352
 - `"files"` (attribute value) 30
 - `"files"` (attribute value)
..... 21, 30, 70, 129, 133, 135
 - `<files>` (element) 384
 - `files` (template) [8671](#)
 - files:
 - `clean` [4](#)
 - `ctan` [4](#)
 - `dtx` [4](#)
 - `install` [4](#)
 - `recover` [4](#)
 - `tds` [4](#)
 - `web` [4](#)
 - `@filetype` (attribute) 263
 - `"final"` (attribute value) 22
 - `<firstname>` (element) 67, 73, 189
 - `firstterm` [8](#)
 - `<firstterm>` (element) 66
 - `@floatstyle` (attribute) 63
 - `flow` [56](#)
 - `\fnote` [112](#)
 - `@fontalign` (attribute) 17
 - `@fontcolor` (attribute) 68
 - `@fontcolor` (attribute) 17, 69
 - `@fontencoding` (attribute) 68
 - `@fontencoding` (attribute) 17
 - `@fontface` (attribute) 68
 - `@fontface` (attribute) 17
 - `fontface` (attribute) [58](#)
 - `@fontfamily` (attribute) 68
 - `@fontfamily` (attribute) 17
 - `@fontseries` (attribute) 68
 - `@fontseries` (attribute) 17
 - `@fontshape` (attribute) 68
 - `@fontshape` (attribute) 17
 - `@fontsize` (attribute) 68
 - `@fontsize` (attribute) 17, 68
 - `"footer"` (attribute value) 330
 - `<footnote>` (element) 66
 - `<foreignphrase>` (element) 66
 - `<form>` (element) 85
 - Formal Public Identifier *see* FPI
 - `<formalpara>` (element) 359
 - FPI 20, **20**
 - `"frag"` (attribute value) 62, 156
 - `fragments` (setting) [4](#)
 - `"framed"` (attribute value) 61, 234, 351
 - `"full"` (attribute value) 65
 - `fullcite` (template) [24](#)
 - `@funcparams` (attribute) 54
 - `<function>` (element) 66
 - `<function>` (element) 17, 57

functions:

chunk [1](#)
 chunkawk [3](#)
 decommentbbl [3](#)

G

' 299
 &LaTeX; 32
 &TeX; 32
   32
 graphicx (package) [11](#)
 <gui*> (element) 181
 <guibutton> (element) 66
 <guilabel> (element) 66
 <guimenu> (element) 66
 <guimenuitem> (element) 66
 <guisubmenu> (element) 66

H

"hand" (attribute value) 51
 "header" (attribute value) 64, 343
 hierarchy [56](#)
 <holder> (element) 25
 @href (attribute) 146
 \hrule [102](#)
 HTML (entity) [118](#)
 html (DTD) [5](#)
 <html:form> (element) 16, 85
 htrim (template) [7891](#)
 \hyphenation [116](#)

I

<ID> (element)
 . 57, 70, 94, 249, 257, 300, 363, 366
 IDREF (local name) 54
 <IDREF> (element) 16, 70, 257, 300
 <IDREFS> (element) 16, 29, 57, 300
 \ifdefined [7](#)
 \IfPackageLoaded [7, 8, 6511](#)
 "ignore" (attribute value) 61
 <imagedata> (element)
 . 53, 63, 67, 217, 351
 imagedata (template) [1099](#)
 imagedatafiles (template) [1121](#)
 imagedataremap (template) [1204](#)
 <imageobject> (element) .. 63, 67, 354
 imageobject (template) [1073](#)
 index (template) [8754](#)
 IndexColumns (counter) [13, 108](#)
 <indexterm> (element) 190, 191
 <info> (element) ... 24, 27, 33, 97, 332
 <info> (start-tag) 20
 info (template) [6](#)
 informalfigure (template) [1046](#)
 <informaltable> (element) 64, 70

<informaltable> (element)
 . 57, 59, 64, 70
 informaltable (template) [156](#)
 informaltableinfo (template) [186](#)
 init (template) [3, 3, 3](#)
 <inlineequation> (element) 66
 inlineequation (template) [4780](#)
 <inlinemediaobject> (element) ... 67
 insfile (template) [8312](#)
 install (file) [4](#)
 invocation (entity) [186](#)
 <issuenum> (element) 74
 issuenum (template) [1167](#)
 "it" (attribute value) 68
 italics (template) [507](#)
 <itemizedlist> (element) .. 60, 67, 69
 <itemizedlist> (element)
 . 48, 56, 57, 59, 60, 163

J

"journal" (attribute value) 72

K

"k" (attribute value) 68
 <keycap> (element) 67

L

"l" (attribute value) 68
 \l@section [25](#)
 \l@subsubsection [27](#)
 @label (attribute) 71
 @label (attribute) 16, 71, 86, 316
 label (attribute) [19](#)
 @Lang (attribute) 401
 langs (variable) 93, [106](#)
 @language (attribute) 61
 @language (attribute)
 . 16, 60, 66, 67, 69, 156, 180
 language (attribute) [40](#)
 "LaTeX" (attribute value) 67
 &LaTeX; (general entity) 32
 LaTeX (entity) [74](#)
 "LaTeX2e" (attribute value) 22
 "LaTeXe" (attribute value) 61
 latexprog (variable) 96
 "length" (attribute value) 69, 175, 201
 lengths:
 columnsep 80, [109](#)
 CPKtabwid [110](#)
 pagelen [14](#)
 licence (variable) 93
 @linenumbering (attribute) .. 156, 234
 <link> (element) 67, 69
 <link> (element) 69, 102
 link (template) [567](#)

`<linkend>` (element) 85
`@linkend` (attribute)
 16, 34, 37, 44, 54, 57,
 60, 65, 66, 69, 70, 103, 173, 204, 219
`linkend` (attribute) [17](#)
`<listitem>` (element) 69, 70
`<listitem>` (element)
 54, 56, 57, 60, 70, 169
`listitem` (template) 169
`<literal>` (element) 67
`<literal>` (element) 17
`<literallayout>` (element) 67
`<literallayout>` (element) 17, 32
`literate-programming` **13**
`loc` (parameter) 239, 336
`EMPTY` 16, 29, 61, 75
`IDREF` 54
`PUBLIC` 20
`SYSTEM` 20
`"lpl"` (attribute value) 22
`ltrim` (template) 237, [7923](#)
`ltxcmds` (package) [6](#)
`ltxdoc@ltxdoc` (class) 45, 48
`"lualatex"` (attribute value) 218
`\LyX` [66](#)

M

`"m"` (attribute value) 68, 339
`\MacroFont` [15](#)
`"maintainer"` (attribute value) 25
`makechapapp` (template) . 135, 136, [6830](#)
`makecommentchar` (template) [7492](#)
`makefence` (template) [7534](#)
`makelisting` (template) 156, [5994](#)
`makelistings` (template) 156
`makepackages` (template) 216
`makeref` (template) [14](#), 299
`makesentinel` (template) [7502](#)
`maketermsentinel` (template) ... [7518](#)
`"manifest"` (attribute value) 46
`manifest` (template) [7434](#)
`marginnote` (package) [12](#)
`<markup>` (element) 67
`markup` (template) [785](#)
`@match` (attribute) 152
`"math"` (attribute value) 51
`maxcolen` (variable) 94, 274
`<mediaobject>` (element) 63
`mediaobject` (template) [1053](#)
`<member>` (element) 46
`metadata` (template) 365, [742](#)
`<methodname>` (element) 67
`<methodparam>` (element) .. 53, 54, 102
`<methodsynopsis>` (element) 211

`methodsynopsis` (template) [648](#)
`mode` (parameter) 239
`<modifier>` (element) 53, 102
`"mono"` (attribute value) 51
`"monospace"` (attribute value) 68
`monospace` (template) [500](#)
`multicite` (template) [16](#)

N

`"n"` (attribute value) 68
`name` (variable) 94, 307
`"new"` (attribute value) 283
`"newpage"` (attribute value) 343
`"news"` (attribute value) 200
`"noat"` (attribute value) 68
`"nodesc"` (attribute value) 65, 201
`"nolit"` (attribute value) 267
`nondeferred` (template) [6368](#)
`<none>` (element) 185
`normtext` (template) [808](#)
`"nosc"` (attribute value) 52
`<note>` (element) 64, 70
`<note>` (element) 57, 59
`noto` (package) [4](#)

O

`"off"` (attribute value) 109, 226
`<olink>` (element) 31, 360
`@omit` (attribute) 17
`omit` (attribute) [66](#)
`"only"` (attribute value) 41, 242
`<option>` (element) 67
`<option>` (element) 17, 57
`"options"` (attribute value) 37, 44
`<orderedlist>` (element) ... 60, 67, 70
`<orderedlist>` (element)
 48, 56, 57, 59, 60, 70
`<orderedlist/listitem>` (element) 70
`<org>` (element) 74
`<orgdiv>` (element) 73, 74
`<orgname>` (element) 73, 74
`orgname` (template) [1218](#)
`orguri` (template) [1050](#)
`@os` (attribute) 22
`"other"` (attribute value) 74
`<otherclass>` (element) 74
`<othername>` (element) 73
`OUTPUT` (entity) [71](#)
`"overhang"` (attribute value) 334

P

`"p"` (attribute value) 65
`package` **10**
`<package>` (element) 67
`"package"` (attribute value) 22

- `<package>` (element) 57
 - `<packages>` (element) 109
 - `packages` (template) [6393](#)
 - `packages` (variable) 108, 109
 - `packages`:
 - `fancyhdr` [5](#)
 - `graphicx` [11](#)
 - `ltxcmds` [6](#)
 - `marginnote` [12](#)
 - `noto` [4](#)
 - `parskip` [10](#)
 - `\PackageWarning` [7100](#), [7105](#), [7254](#)
 - `"page"` (attribute value) 70
 - `"page"` (attribute value) 57
 - `pagelen` (length) 14
 - `<pagenums>` (element) 74
 - `pagenums` (template) [1123](#)
 - `"paper"` (attribute value) 72
 - `<para>` (element) 59
 - `<para>` (element)
 - 27, 32, 36, 50, 52, 56, 57, 59, 60, 65
 - `<parameter>` (element) 67
 - `"parameter"` (attribute value) 67
 - `<parameter>` (element)
 - 17, 53, 54, 57, 102, 201
 - `@parameter` (attribute) 54
 - `Parameter Entity` *see* PE
 - `parameters`:
 - `armour` [30](#)
 - `dest` 239
 - `loc` 239, 336
 - `mode` 239
 - `pkg` 239
 - `parawarn` (template) [178](#)
 - `"paren"` (attribute value) 65
 - `parskip` (package) [10](#)
 - `<part>` (element) 70
 - `<part>` (element) 21, 27,
 - 30, 53, 55, 56, 58, 70, 96, 132, 218
 - `<part>` (start-tag) 21, 23
 - `passthru` (template) [772](#)
 - `pdf` (setting) [4](#)
 - `"pdflatex"` (attribute value) 58
 - `PE` **85**, 87
 - `<personname>` (element) 24, 73
 - `<phrase>` (element) 67
 - `PI` 62, **389**, 393
 - `"pkg"` (attribute value) 239
 - `pkg` (parameter) 239
 - `"pl"` (attribute value) 22
 - `plaintext` (template) [15](#)
 - `pool` **56**
 - `"postpackage"` (attribute value) 50, 55
 - `precite` (template) [224](#)
 - `<preface>` (element) 58
 - `"prepackage"` (attribute value)
 - 50, 55, 241
 - `<prepackage>` (element) 216
 - `prepost` (variable) 93
 - `prepostno` (variable) 221
 - `<procedure>` (element) 70
 - `<procedure>` (element)
 - 49–52, 56, 57, 65, 109
 - `<procedure/step>` (element) 70
 - `"proceedings"` (attribute value) ... 72
 - `Processing Instruction` *see* PI
 - `\ProcessOptions` [1910](#)
 - `<productname>` (element) 67
 - `<productnumber>` (element) 325
 - `<programlisting>` (element) 70
 - `<programlisting/>` (empty element) 31
 - `<programlisting>` (element)
 - 16, 17, 27, 30, 31,
 - 38, 49, 57, 59–61, 70, 190, 198,
 - 217, 230, 273, 279, 283, 288, 385
 - `\providecommand` [520](#), [522](#)
 - `pubaddress` (template) [971](#)
 - `PUBLIC` (local name) 20
 - `<publisher>` (element) 74
 - `publisher` (template) [925](#)
 - `<publishername>` (element) 74
 - `publishername` (template) [941](#)
 - `"py"` (attribute value) 22
- Q**
- `<quote>` (element) 67
 - `<quote>` (element) 16
 - `quoted` (template) [516](#)
- R**
- `"r"` (attribute value) 68
 - `"rc"` (attribute value) 22
 - `readme` (template) [7389](#)
 - `readme` (variable) 93
 - `recover` (file) [4](#)
 - `<refsection>` (element) 49
 - `@relation` (attribute) 73
 - `<releaseinfo>` (element) 25
 - `<releaseinfo>` (element) 35, 321
 - `@remap` (attribute) 23, 61
 - `@remap` (attribute) 50, 51,
 - 54, 65, 108, 109, 143, 156, 175,
 - 182, 234, 307, 314, 334, 343, 354
 - `<remark>` (element) 65
 - `<remark>` (element) 16, 26, 59, 154
 - `repeatarg` (template) [8046](#)
 - `<replaceable>` (element) 68
 - `<replaceable>` (element) 267

`<repleaceable>` (element) 270
`@reuse` (attribute) 16, 29, 153, 154
`<revdescription>` (element) 36
`<revhistory>` (element) .. 19, 22, 23, 26
`<revhistory>` (element) ... 23, 76, 115
`@revision` (attribute) 22, 26
`<revision>` (element) 19, 22, 76
`@revision` (attribute)
..... 16, 19, 33, 36, 51, 65, 76
`revision` (template) [4732](#)
`rewrap` (template) [6758](#)
`<role>` (element) ... 25, 27, 44, 69, 242
`@role` (attribute) 16, 17,
25, 28, 29, 34, 41, 51, 57, 64, 65,
69, 86, 109, 117, 132, 149, 150,
153, 156, 167, 175, 177, 178,
180, 187, 192, 201, 240, 257,
304, 309, 312, 315, 330, 331, 343
`role` (attribute) [30](#)
`"roman"` (attribute value) 68
`<row>` (element) 64, 343, 401
`row` (template) [710](#)
`@rowsep` (attribute) 64, 343
`"ruled"` (attribute value) 343

S

`"sans"` (attribute value) 51, 68
`"sc"` (attribute value) 68
`<screen>` (element) 17
`"script"` (attribute value) 22
`secnumdepth` (counter) [106](#)
`secondary` (template) [4204](#)
`<sect1>` (element) 60
`<sect1>` (element) 26, 27, 30,
32, 44, 45, 58–60, 139, 250, 256, 363
`<sect2>` (element) 60
`<sect2>` (element) 26, 58–60, 140
`<sect3>` (element) 60
`<sect3>` (element) 58–60
`<section>` (element) 58, 141, 306
`@security` (attribute) 19, 23
`@security` (attribute) 19, 33, 76
`<seg>` (element) 33, 40, 41, 52, 109, 239
`<seglistitem>` (element) 17, 33, 40,
41, 51, 52, 108, 109, 221, 227, 239
`seglistitem` (element)
[5574](#), [5598](#), [5657](#), [5701](#), [5747](#), [5770](#)
`<segmentedlist>` (element) . 40, 45, 50
`<segtitle>` (element) 41, 45
`sentinel` (variable) 95, 264
`series` (template) [1226](#)
`"serif"` (attribute value) 51
settings:
dependencies [4](#)

fragments [4](#)
pdf [4](#)
shortcuts [4](#)
values [4](#)
`setup` (template) [10](#)
`shield` (variable) 92, 328
`"short"` (attribute value) 57
`"shortauthor"` (attribute value) ... 65
`shortcuts` (setting) [4](#)
`"shorttitle"` (attribute value) ... 65
`@show` (attribute) 283
`<sidebar>` (element) 64, 70
`<sidebar>` (element) 57, 59
`silcommentchar` (template) [7564](#)
`silsentinel` (template) [7594](#)
`siltermcommentchar` (template) .. [7578](#)
`<simplelist>` (element) 46
`"sl"` (attribute value) 68
`\SMC` [67](#)
`@spacing` (attribute) 39, 60
`@spanname` (attribute) 64
`"sponsor"` (attribute value) 25
`<book>` 20, 24
`<info>` 20
`<part>` 21, 23
`startappcomment` (template) [1782](#)
`@startinglinenumber` (attribute) ... 62
`@startinglinenumber` (attribute) ...
..... 61, 62, 156, 236
`starttag` (template) [7548](#)
`@status` (attribute) 22
`<status>` (element) 22
`<step>` (element) 70
`<step>` (element) 50–
53, 55, 57, 65, 103, 109, 216, 221
`"strong"` (attribute value) 177
`"sty"` (attribute value) .. 22, 51, 53, 109
`@style` (attribute) 343
`"stypackages"` (attribute value) ...
..... 37, 39, 41, 52
`\subsubsection` [31](#)
`<subtitle>` (element) 59
`<subtitle>` (element) 59, 73, 126
`<surname>` (element) 67, 73, 189
`"switch"` (attribute value) 175
`"symbol"` (attribute value) 51
`SYSTEM` (local name) 20
`<systemitem>` (element) 68

T

`\tabcolsep` [21](#)
`<table>` (element) 63
`<table>` (element) 57, 59
`table` (template) [10](#)

- tabletitle (template) [120](#)
- tabstart (template) [281](#), [332](#)
- <tag> (element) [56](#), [68](#)
- <tag/> (empty element) [8](#)
- <tag> (element) [87](#)
- <targetptr> (element) [32](#)
- <task> (element) [64](#)
- <task> (element) [59](#), [65](#)
- <tbody> (element) ... [63](#), [332](#), [333](#), [343](#)
- tbody (template) [608](#)
- tds (file) [4](#)
- "template" (attribute value) [201](#)
- templates:
 - abstract [1204](#)
 - addindent [273](#), [7957](#)
 - address [1209](#)
 - appendix [249](#)
 - avoidverb [175](#), [179](#), [201](#), [7601](#)
 - biblioallcite [310](#)
 - biblioauthed [524](#)
 - biblioauthgroup [497](#)
 - biblioauthsolo [542](#)
 - biblioauthterm [611](#)
 - bibliocorpauth [618](#)
 - bibliodata [381](#)
 - biblioentry [413](#)
 - bibliofile [327](#)
 - bibliography [298](#)
 - bibliolist [296](#)
 - bibliomisc [455](#)
 - biblionewpage [321](#)
 - biblioorgs [655](#)
 - biblioprintbiblatex [338](#)
 - biblioprintbibtex [360](#)
 - bibliosets [460](#)
 - biblioshorttitles [794](#)
 - bibliosources [799](#)
 - bibliosubtitles [720](#)
 - biobotitles [671](#)
 - breakline [94](#), [7978](#)
 - casestyle [6339](#)
 - chapter [249](#)
 - checkpackages [256](#), [7175](#)
 - citeauthors [117](#)
 - citedetail [29](#)
 - citelink [273](#)
 - citeother [163](#)
 - citeshort [63](#)
 - citesurnames [153](#)
 - citetitle [34](#)
 - citeunits [240](#)
 - colprefix [334](#), [356](#)
 - colsettings [334](#), [489](#)
 - colsuffix [334](#), [571](#)
 - compspace [8038](#)
 - confdates [1065](#)
 - confgroup [994](#)
 - conforgaddress [1055](#)
 - confsponsor [1045](#)
 - conftitle [999](#)
 - copyright [8064](#)
 - corefs [4725](#)
 - coverage [1232](#)
 - dates [1081](#)
 - db:abstract [1076](#)
 - db:acknowledgements [2046](#)
 - db:acronym [486](#), [3273](#)
 - db:affiliation [1428](#)
 - db:anchor [404](#)
 - db:annotation ... [2353](#), [2563](#), [5131](#)
 - db:appendix [1725](#)
 - db:author [1394](#)
 - db:biblioref [10](#), [694](#)
 - db:blockquote [2234](#)
 - db:chapter [1625](#)
 - db:citetitle [3397](#)
 - db:classname [3460](#)
 - db:cmdsynopsis [1113](#)
 - db:code [3435](#)
 - db:command [582](#), [3508](#)
 - db:constraintdef [2511](#)
 - db:declopts [5155](#)
 - db:email [3613](#)
 - db:emphasis [3597](#)
 - db:envar [3618](#)
 - db:errortext [3647](#)
 - db:exceptionname [267](#), [3655](#)
 - db:filename [3663](#)
 - db:firstterm [3672](#)
 - db:footnote [523](#), [3680](#)
 - db:foreignphrase [3685](#)
 - db:formalpara [206](#), [3258](#)
 - db:function [3727](#)
 - db:guibutton [3833](#)
 - db:guiicon [3802](#)
 - db:guilabel [3827](#)
 - db:guimenu [3741](#)
 - db:guimenuitem [3784](#)
 - db:guisubmenu [3766](#)
 - db:honorific [1420](#)
 - db:important [3841](#)
 - db:indexterm [3846](#)
 - db:info [1266](#)
 - db:itemizedlist [2873](#)
 - db:keycap [3856](#)
 - db:link [3883](#)
 - db:listitem [3171](#)
 - db:literal [2225](#)

db:literallayout	457, 3935
db:markup	4222
db:methodname	3981
db:note	3992, 4006
db:olink	454
db:option	4021
db:orderedlist	2930
db:package	4061
db:para	312, 354, 2060
db:part	1476
db:personname	4097
db:pi	5118
db:preface	1450
db:primary	4136
db:productname	4143
db:programlisting	229, 2623
db:quotes	4154
db:releaseinfo	1446
db:remark	2570
db:replaceable	4167
db:sect1	386, 1850
db:sect2	1919
db:sect3	1956
db:sect4	1984
db:section	2012
db:see	4212
db:simplelist	3125
db:superscript	4217
db:tag	592, 4228
db:task	4518
db:term	281, 3209
db:title	52, 1284, 2030, 4534
db:token	4539
db:type	4545
db:typecol	4642
db:uri	513, 4650
db:userinput	2867
db:variablelist	270, 3083
db:varname	4675
db:warning	151, 2264
db:xref	528, 4802
default	777, 5358
delogify	8271
detex	966
dodate	1084
dologo	376, 944
done	1237
dtxttext	214, 8207
edition	1172
end	1034, 9144
endtag	7556
entrybody	902
entryheadfoot	786
extinc	2809
figure	1014
files	8671
fullcite	24
htrim	7891
imagedata	1099
imagedatafiles	1121
imagedataremap	1204
imageobject	1073
index	8754
info	6
informalfigure	1046
informaltable	156
informaltableinfo	186
init	3, 3, 3
inlineequation	4780
insfile	8312
issuenum	1167
italics	507
link	567
listitem	169
ltrim	237, 7923
makechapapp	135, 136, 6830
makecommentchar	7492
makefence	7534
makelisting	156, 5994
makelistings	156
makepackages	216
makeref	14, 299
makesentinel	7502
maketermsentinel	7518
manifest	7434
markup	785
mediaobject	1053
metadata	365, 742
methodsynopsis	648
monospace	500
multicite	16
nondeferred	6368
normtext	808
orgname	1218
orguri	1050
packages	6393
pagenums	1123
parawarn	178
passthru	772
plaintext	15
precite	224
pubaddress	971
publisher	925
publishername	941
quoted	516
readme	7389
repeatarg	8046
revision	4732

rewrap [6758](#)
 row [710](#)
 secondary [4204](#)
 series [1226](#)
 setup [10](#)
 silcommentchar [7564](#)
 silsentinel [7594](#)
 siltermcommentchar [7578](#)
 startappcomment [1782](#)
 starttag [7548](#)
 table [10](#)
 tablettitle [120](#)
 tabstart [281](#), 332
 tbody [608](#)
 term 168
 text [635](#), [5313](#)
 textobject [1228](#)
 tgroup [199](#)
 tgroupend [278](#)
 tgroupfakefootnotes [238](#)
 thead [600](#)
 token [865](#)
 typeset [829](#)
 vars [9](#)
 version [989](#)
 volumenum [1155](#)
 <term> (element) 60, 169, 175, 361
 term (template) 168
 "test" (attribute value) 343
 "TeX" (attribute value) 67
 &TeX; (general entity) 32
 "text" (attribute value) 57, 65
 text (template) [635](#), [5313](#)
 <text()> (element) 376
 <textobject> (element) 330, 354
 textobject (template) [1228](#)
 <tfoot> (element) 63, 332, 345
 <tgroup> (element)
 43, 64, 103, 331, 332, 334
 tgroup (template) [199](#)
 tgroupend (template) [278](#)
 tgroupfakefootnotes (template) .. [238](#)
 @tgroupstyle (attribute) 334
 "thead" (attribute value) 336
 <thead> (element) 332, 341, 345
 thead (template) [600](#)
   (general entity) 32
 thisdoc (variable) 93, 217
 thisdocfullname (variable) 249
 thisdocfulltype (variable) 249
 thisdocid (variable) 249
 thisdocname (variable) 249
 thisdoctype (variable) 249
 thisversion (variable) [41](#)

<title> (element) 24, 59
 "title" (attribute value) 65, 66
 <title> (element) 24, 26, 35, 58, 59,
 64, 71, 73, 126, 170, 300, 330, 332
 <titleabbrev> (element) 73
 tocdepth (counter) [107](#)
 <token> (element) 190
 @token (attribute) 159
 token (template) [865](#)
 \tubhideheight [56](#)
 \tubreflect [42](#)
 "TUG'95" (attribute value) 75
 <type> (element) 68, 69
 <type> (element) 17, 32, 69, 200
 @type (attribute) 200
 typeset (template) [829](#)

U

@units (attribute) 16, 65
 units (attribute) [11](#)
 "up" (attribute value) 68
 <uri> (element) 69
 "uri" (attribute value) 74
 <uri> (element) 102
 usemakefile-comment (comment) [1035](#)
 <userinput> (element) 160, 234
 @userlevel (attribute) 22, 30
 <userlevel> (element) 22
 @userlevel (attribute)
 30, 50, 67, 109, 190, 226, 263

V

values (setting) [4](#)
 "variable" (attribute value) 201
 <variablelist> (element) 60
 <variablelist> (element)
 56, 57, 59, 60, 168, 361, 363
 variables (variable) [4](#)
 variables:
 appdir 93
 armour 307
 attribmatch [5408](#)
 attvalmatch [5438](#)
 attvalptrmatch [5482](#)
 commentchars 95, 263
 cpdir 93
 defaultmatch [5508](#)
 dependencymatch [5513](#)
 docname 94
 elementmatch [5387](#)
 elements 217, 221, [5539](#)
 langs 93, 106
 latexprog 96
 licence 93

maxcodelen 94, 274
 name 94, 307
 packages 108, 109
 prepost 93
 prepostno 221
 readme 93
 sentinel 95, 264
 shield 92, 328
 thisdoc 93, 217
 thisdocfullname 249
 thisdocfulltype 249
 thisdocid 249
 thisdocname 249
 thisdoctype 249
 thisversion [41](#)
 variables [4](#)
 <varlistentry> (element) 69
 <varlistentry> (element)
 54, 57, 60, 69, 169, 219
 <varname> (element) 69
 <varname> (element) 17, 57, 86
 @varname (attribute) 67
 vars (template) [9](#)
 @vendor (attribute) 23
 @vendor (attribute) 51
 @version (attribute) 19, 22
 @version (attribute)
 16, 19, 33, 36, 41, 65, 76, 109
 version (attribute) [38](#)
 version (template) [989](#)
 <volumenum> (element) 74
 volumenum (template) [1155](#)

W

<warning> (element) 64, 70
 <warning> (element) 57, 59
 web (file) [4](#)
 "widest" (attribute value) 339
 @width (attribute) 63, 352
 <wordasword> (element) 69
 @wordsize (attribute) 49, 61, 64, 68
 @wordsize (attribute)
 64, 147, 156, 234, 331, 336

X

\XeLaTeX [65](#)
 "xelatex" (attribute value)
 55, 58, 109, 216, 218

\XeTeX [58](#)
 @xlink:href (attribute) 30, 62, 71
 @xlink:href (attribute) 30,
 31, 36, 38, 61, 67, 113, 156, 182,
 184, 200, 230, 249, 283, 307, 333
 @xlink:role (attribute) 23, 41
 @xlink:role (attribute) .. 41, 112, 242
 @xlink:show (attribute) 156
 "XML" (attribute value) 66, 69
 @xml:base (attribute) 23
 @xml:id (attribute) 21, 30
 <xml:id> (element) 66
 @xml:id (attribute) 16,
 18, 21, 23, 24, 29, 30, 34, 37, 39,
 40, 44–46, 54, 56–58, 60, 65, 66,
 69, 71, 76, 94, 128, 129, 143, 146,
 154, 156, 168, 170, 183, 234, 330
 @xml:lang (attribute) 49, 66, 67
 <xref> (element) 69
 <xref> (element) 16, 53, 54, 57, 204, 219
 @xref (attribute) 159
 @xreflabel (attribute) 70, 71
 <xreflabel> (element) 27
 @xreflabel (attribute)
 16, 23, 29, 66, 70,
 71, 80, 86, 149, 156, 200, 201, 216
 @xrefstyle (attribute) 70
 @xrefstyle (attribute) 57, 65, 66
 "xsl" (attribute value) 30
 <xsl:apply-templates> (element) .
 159, 267
 <xsl:include> (element) 299
 <xsl:otherwise> (element) .. 132, 142
 xsl:output (element) [18](#)
 <xsl:result-document> (element) . 92
 <xsl:stylesheet> (element) 91
 xsl:stylesheet (element) [3](#)
 <xsl:value-of> (element) 159
 "XSLT" (attribute value) 66, 69

Y

<year> (element) 16, 25
 @YYYY-MM-DD (attribute)
 16, 26, 35, 73–76
 YYYY-MM-DD (attribute) [7](#)
 @YYYY-MM-DD-from (attribute) 75
 @YYYY-MM-DD-to (attribute) 75