chemstyle — Writing chemistry with style *

Joseph Wright †

Released 2007/10/31

Abstract

The chemstyle package provides a "one-stop shop" for setting up formatting of LaTeX documents following the editorial policies of various chemical journals. It provides a number of handy chemistry-related commands, and loads several support packages to aid the chemist.

Contents

*This file describes version v1.1h, last revised 2007/10/31.

†E-mail: joseph.wright@morningstar2.co.uk

Option	Journals using this style
none	Not applicable
angew	*Angew. Chem., Chem. Eur. J.*
jomc	*J. Organomet. Chem., Coord. Chem. Rev.*
ic	*Inorg. Chem.*
jacs	*J. Am. Chem. Soc.*
jcp	*J. Phys. Chem. A, J. Phys. Chem. B*
orglett	*Org. Lett.*
rsc	*Chem. Commun., Org. Biomol. Chem. Dalton Trans.*
tetlett	*Tetrahedron, Tetrahedron Lett.*

Table 1: Styles provided by `chemstyle`

1 Introduction

The aim of `chemstyle` is to provide a quick method to set up various document parameters (such as caption formatting), simply by specifying the model journal. The package has also been designed to allow rapid addition of new journal styles. Each style definition is a separate file, and new styles can be added very readily. `chemstyle` has grown out of the `rsc` package, which had a similar aim but was much more limited (and less robustly implemented). The `chemstyle` package is also designed with the use of `biblatex` in mind: the `rsc` package is closely bound to traditional BibTeX use.

As a successor to the `rsc` package, `chemstyle` provides a range of chemistry-related additional macros. The set provided here is an extended version of those provide by `rsc`. Everything that can be done using the `rsc` LaTeX package is therefore possible using the `chemstyle` package.

The formatting system provided by `chemstyle` are intended for writing a variety of documents. Thus the stylistic changes made by the package do not seek to reproduce the appearance of printed journal articles. The package aims to be suitable for use in preparing drafts of papers, but also for writing reports, theses and so on.

2 Style options for `chemstyle`

`chemstyle` should be loaded with a package option specifying which journal style to follow. Currently, `chemstyle` is aware of a the styles listed in Table 1. New styles can be developed by creating a new file modelled on the existing definitions; `chemstyle` will automatically search for correctly-named styles. The style files provided with `chemstyle` have been derived from current practice in the target journals. It is not always easy to pick the correct stylistic settings from (sometimes inconsistent) real-world examples. The package author welcomes feedback on the styles provided.

The `none` style is notable as it is not based on a journal. Instead this is a minimal style, which provides the additional commands without making formatting changes. It is also the default style if no option is given. This

document has been compiled implicity using the `none` option, for example.

2.1 Creating styles

The process of creating a new styles for `chemstyle` is intended to be relatively easy. New styles should be saved as files with the extension `.jdf` (standing for for Journal Definition File), and should be saved somewhere in the path searched by TeX.[1] The definition files included in the package should provide a guide to the basic options available for producing new styles. Arbitrary TeX commands can be included, if they are necessary for a particular style. For example, other packages can be loaded in `\usepackage`.

The maintainer of `chemstyle` is happy to add new styles to the package, either by contribution by users or on request (when he has sufficient time!). If you have a new style to add (or corrections to an existing one), please contact the package author.

2.2 Package options

As well as the various journal styles provided with the package, a number of options are recognised to alter compatibility with other packages and to give the desired output. `chemstyle` will also pass options through to the `chemscheme` package, which is used to provide support for schemes. Users should consult the `chemscheme` documentation for the options applicable.

Currently, `chemstyle` has options `nonotes`, `nophrases`, `SIunits` and `xspace`. These are explained at the appropriate places within this document. All are hopefully relatively clearly named.

3 Naming of the references section

`chemstyle` alters the naming of the references section of a document. By default, `chemstyle` alters the value of `\bibname` or `\refname` (as appropriate) to the form of words chosen by the target journal for the "Notes and References" section.

The "Notes and References" naming commands are language-aware, *via* the `babel` interface. Currently, `chemstyle` includes appropriate labels for babel languages `english`, `UKenglish`, `ngerman` and `french`.[2] Other languages can be added if appropriate wordings are provided to the author.

The `chemstyle` packages recognises the `nonotes` option, which affects how the Notes and Referencessection is headed. With the `nonotes` option, the "Notes" part of the section name is omitted, and only the "Notes and References" phrase is output.

[1] Depending on your TeX distribution, you may need to rebuild your file database after creating a new style. For MikTeX users, this can be done graphically or by typing `mpm -update-db` at the command line; for TeXLive, run `texhash`.

[2] The `french` option for `babel` causes a clash with `unitsdef` concerning the command `\fg`. `babel` uses this for *guillemets*, while `unitsdef` uses it as an abbreviation for `\femtogram`. The `chemstyle` package prevents the clash, and leaves the `babel` definition intact.

4 Additional macros

4.1 Additional units

Both the `unitsdef` and `SIunits` packages provide a great number of easy to use unit commands, and handles spacing between numbers and units very well. However, a few useful units seem to be missing, and are provided here. By default, `chemstyle` uses the `unitsdef` package to achieve this, but using the `SIunits` option will use that package instead. The most obvious of these is `\Hz`, which simply gives Hz. The command `\mmHg` has a non-breaking thin space, leading to mm Hg. Two related commands are given for concentration: `\molar` gives mol dm^{-3}, whilst `\Molar` gives M. Finally, the command `\cmc` is provided for generating cm^3. When using `unitsdef`, all of the new macros space correctly with numbers, so inputting `10\cmc` gives 10 cm^3, with a non-breaking space. Users of `SIunits` will be familiar with the approach used by that package.

`\Hz` `\mmHg` `\molar` `\Molar` `\cmc`

4.2 The standard state symbol

Related to the above, but not exactly a unit is the `\standardstate` command.[3] This generates the tricky $^{\ominus}$ symbol: `$\Delta S\standardstate$` generates $\Delta S^{\ominus}$. Note that this is safe in either text or maths mode.

`\standardstate`

4.3 Alkyl radicals

There are a few alkyl radicals that come up all of the time. No one seems to have put these into a package, so they are provided here. As you would expect, `\iPr` gives iPr, `\iBu` gives iBu and `\tBu` gives tBu, and so on. The style of the output depends on the journal style specified; most journals seem to favour one version of the abbreviation. When `chemstyle` is loaded using the `xspace` option, the `xspace` package is automatically used to add space after the command names, so that `\iPr group` will result in "iPr group" being typeset.[4]

`\iPr` `\iPr` `\nBu` `\iBu` `\sBu` `\tBu`

4.4 Latin phrases

The various Latin phrases commonly used in chemistry are made available as the obvious commands. By altering the definition of `\latin`, this allows ready switching from italic to Roman typesetting. Notice that `\etc`, `\ie` and `\eg` are aware of trailing periods, and so doubling-up should not occur. Once again, these macros use `xspace` when given as a package option to handle automatic addition of spaces after these phrases. For American journals, where it is obligatory to follow "*e.g.*" and "*i.e.*" with a comma, the package provides a mechanism for handling this automatically. Thus, when using an appropriate journal style, `\eg`, `\eg.` and `\eg,` will all result in typesetting "*e.g.*,".

`\latin` `\etc` `\eg` `\ie` `\etal` `\invacuo`

The `\etc` and `\etal` commands are set up on the assumption that they come at the end of a sentence. Hence the spacing after these will default to an inter-sentence space. If you desire an inter-word space, use the normal methods

[3] The `\standardstate` macro is only defined if the user does not have their own version.

[4] Reader of the source of this document will see that this document has been compiled *without* the `xspace` option!

`\etc\ more text`, `\etc~more text`, *etc*. If this is a big issue, the package author may look at it again.

The definitions of all of the phrases are designed not to overwrite any given by the user *in the preamble*. So, if you have your own `\latin` macro, it will be used even if you load `chemstyle`. If you encounter any problems, try loading the package with the `nophrases` option; this option prevents the package even trying to define any of the phrase macros.

`\latinemphon` `\latinemphoff`

The formatting of Latin phrases can be altered using the two macros `\latinemphon` and `\latinemphoff`, which switch the use of `\emph` for Latin phrases on and off. These macros are mainly intended for the use in journal style files, but may be more generally useful.

5 Additional information

5.1 Advice for users of `rsc` and `achemso`

The `chemstyle` package is intended as a replacement for the `rsc` package. As such, it covers almost everything the the `rsc` LaTeX package does, and more. Users of `rsc` are strongly encouraged to update to using `chemstyle`. The bibliography styles provided by `achemso` and `rsc` will continue to be required, of course. Migration of these styles to `biblatex` is an on-going project.

5.2 Interactions with other packages

The `chemstyle` package has been designed to avoid, as far as possible, clashes with other packages. The package requires the presence of the standard `graphicx` and `varioref` packages. If these packages need to be loaded with explicit options, this should be done *before* loading `chemstyle`. The `chemscheme` package is also needed, as it provides the floating `scheme` environment essential in synthetic chemistry documents.

5.3 Captions above floats

The `scheme` float type is generated using the `float` package. This has the side-effect that the placement of captions for floats does not depended on where the `\caption` command comes inside the floating environment.[5] If you wish to alter the placement of captions, the mechanism of the underlying `float` package will be needed. This requires the use of `\floatstyle` to load the desired settings, followed by `\restylefloat` to apply the settings. This is most usually done with tables.

[5] Normally this is a good thing.

```
\begin{table}[h]
\caption{A caption above the float contents}
The float contents
\end{table}
\floatstyle{plaintop}
\restylefloat{table}
\begin{table}[h]
\caption{A second caption above the float contents}
The second float contents
\end{table}
```

The float contents

Table 2: A caption above the float contents

Table 3: A second caption above the float contents

The second float contents

6 A template for chemical articles

This is a very simple template for chemistry-related documents. Hopefully it contains a few extra hints for getting well-formatted documents quickly. For simplicity, the template assumes that the user is writing a thesis for a U.K. university. Hence it uses U.K. defaults and RSC-based styling.

The versatile KOMA-SCRIPT bundle provides more advanced versions of the standard document classes. If you want paragraphs separated out, with no indents (a common style for theses), add the `parskip` option to the font and paper size ones given here.

```
1 \documentclass[fontsize=10pt,paper=a4]{scrreprt}
```

The `geometry` package allows the users to alter page layout with ease; much better than trying to hack the raw LaTeX system.

```
2 \usepackage{geometry}
```

The `babel` system is loaded to sort out hyphenation and so on, and could be useful if there are any foreign-language quotes.

```
3 \usepackage[english,UKenglish]{babel}
```

Load `chemstyle` (of course) to give not only some easy formatting, but also to automatically provide a float type for schemes, thanks to the `chemscheme` package. This also loads either `chemcompounds` or (optionally) `bpchem` to track compound numbers. Using the `xspace` option automatically adds space after the various abbreviations.

```
4 \usepackage[rsc,xspace]{chemstyle}
```

The `mhchem` package provides the `\ce` command for rapidly typesetting formulas, so that you can type `\ce{H2SO4}` and get H_2SO_4.

```
5 \usepackage[version=3]{mhchem}
```

Greek letters should be italic if used for variables, but upright (Roman) if used otherwise. So the standard LaTeX `$\delta$` is fine for typesetting an NMR shift (*e.g.* $\delta = \ldots$ ppm.), but not for a σ-bond or a π-complex. The `upgreek` package provides commands such as `$\upsigma$` and `$\uppi$` which are ideal for this.

```
\usepackage{upgreek}
```

The default LaTeX table formatting is not very good. The `booktabs` package does things properly, and has good advice in the manual. A highly-recommended package for those interested in formatting (*i.e.* all (La)TeX users!).

```
\usepackage{booktabs}
```

The `microtype` package improves formatting when used with the pdfTeX engine. By giving the `final` option, it is active even when using `draft` as a global option.

```
\usepackage[final]{microtype}
```

Using `notes2bib` allows the user to automatically add notes to the bibliography from within the document body. So you can put `\bibnote{A note}` in the source, and this will move into the References section without any further effort. The `footnotes` option means that footnotes do the same.

```
\usepackage[footnotes]{notes2bib}
```

Finally, the `hyperref` package makes headings, citations and so on into hyperlinks.

```
\usepackage[colorlinks]{hyperref}
```

The bulk of the document can then start.

```
\begin{document}
\end{document}
```

7 The package code

7.1 Setup code

First of all, the package identifies itself and loads other packages needed to function. Loading `chemscheme` is deferred until later, to allow processing of options at the correct point.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{chemstyle}
  [2007/10/31 v1.1h Writing chemistry with style]
```

Packages that are needed under all circumstances are loaded here.

```
\RequirePackage{graphicx,varioref,caption}
```

7.2 Alkyl radicals

`\cst@emph` `\cst@hyph` `\cst@super`

Alkyl radical abbreviations are produced using the macro `\cst@radical`. This uses a series of formatting commands, which have default values provided here. The names are hopefully pretty self-explanatory.

```
\let\cst@emph\relax
\def\cst@hyph{}
\let\cst@super\relax
\newif  \ifcst@prefix   \cst@prefixtrue
```

\cst@radical The \cst@radical macro does the hard work of declaring each abbreviation. To ensure there is no unexpected clash of names, \newcommand is used first to check for any problems. The commands are made robust so they can (hopefully) be used anywhere.

```
\def\cst@radical#1#2{%
  \expandafter\newcommand\expandafter{\csname #1#2\endcsname}{}%
  \expandafter\DeclareRobustCommand\expandafter{%
    \csname #1#2\endcsname}{%
    \ifcst@prefix
      \cst@super{\cst@emph{#1}}\cst@hyph#2%
    \else
      #2\cst@hyph\cst@super{\cst@emph{#1}}%
    \fi%
    \cst@xspace
  }
}
```

With a mechanism in place, the abbreviations are declared. The format of the resulting output will depend upon the configuration file used.

```
\cst@radical{n}{Pr}%
\cst@radical{i}{Pr}%
\cst@radical{n}{Bu}%
\cst@radical{i}{Bu}%
\cst@radical{s}{Bu}%
\cst@radical{t}{Bu}%
```

7.3 Option handling

Firstly, the "pass through" options are handled, as these will be sent through to chemscheme. The package options for chemstyle are also handled here.

```
\newif  \ifcst@notes    \cst@notestrue
\newif  \ifcst@phrases  \cst@phrasestrue
\newif  \ifcst@si       \cst@sifalse
\newif  \ifcst@xspace   \cst@xspacefalse
\DeclareOption{ch}{\ExecuteOptions{chapter}}
\DeclareOption{chapter}%
  {\PassOptionsToPackage{chapter}{chemscheme}}
\DeclareOption{chemcompounds}%
  {\PassOptionsToPackage{chemcompounds}{chemscheme}}
\DeclareOption{nonotes}{\cst@notesfalse}
\DeclareOption{nophrases}{\cst@phrasesfalse}
\DeclareOption{siunits}{\cst@sitrue}
\DeclareOption{SIunits}{\cst@sitrue}
\DeclareOption{xspace}{\cst@xspacetrue}
```

In order to handle the loading of style definitions correctly, a bit of care is needed. The package options need to be loaded *before* chemscheme, but the style definitions need to be loaded after chemscheme. This is achieved by saving the options and looking at them again once the package loading is done. Notice that only one style definition can be used, so this is taken care of with a warning if needed.

```
\let\cst@option\relax
\DeclareOption*{%
```

```
  \ifx\cst@option\relax\else
    \PackageWarning{chemstyle}
      {Extra option \cst@option\space ignored}
  \fi
  \edef\cst@option{\CurrentOption}
}
```

If `SIunits` is already loaded, then the option has to be set.

```
\@ifpackageloaded{SIunits}
  {\ExecuteOptions{siunits}}
  {\@ifpackageloaded{siunits}
    {\ExecuteOptions{siunits}}
    {}}
\ProcessOptions*
\ifcst@si
  \@ifpackageloaded{unitsdef}
    {\PackageWarning{chemstyle}{%
      You have set the SIunits option,\MessageBreak
      but have already loaded unitsdef!\MessageBreak
      unitsdef will be used for extra unit macros}
    \cst@sifalse}
    {\AtBeginDocument{%
      \@ifpackageloaded{siunits}%
        {}
        {\RequirePackage{SIunits}}%
      }
    }
\else
```

The `\fg` macro is defined in French as a *guillemet* sign, and so to avoid a clash a bit of patching is needed. A check to see if `babel` has already defined `\fg`. If it has, the definition is saved and deleted.

```
  \ifx\fg\@undefined
  \else
    \ifx\fg\relax
    \else
      \let\cst@fg\fg
      \let\fg\@undefined
    \fi
  \fi
  \RequirePackage{unitsdef}
```

If `\cst@fg` exists, it is restored here, otherwise `\fg` is undefined. At the beginning of the document, `\fg` is defined as an abbreviation for a femtogram if it is not being used by `babel`. This way there is not a problem if `babel` is loaded after `chemstyle`.

```
  \ifx\cst@fg\@undefined
    \let\fg\@undefined
  \else
    \let\fg\cst@fg
  \fi
  \AtBeginDocument{%
    \ifx\fg\@undefined
      \newcommand{\fg}{\femtogram}
    \fi
```

```
  }
\fi
\RequirePackage{chemscheme}
```

7.4 Extra units and related commands

\cubiccentimeter
\cmc
\Hz
\Molar
\molar
\mmHg
\mol

A few additional unit types are provides, which `unitsdef` and `SIunits` do not provide. Most of these require different set up for the two supported packages.

```
\ifcst@si
  \newcommand{\cubiccentimeter}{\centi\metre\cubed}
  \newcommand{\Molar}{\textsc{m}}
  \newcommand{\molar}{\mole\usk\deci\metre\rpcubed}
  \newcommand{\mmHg}{\milli\metre~Hg}
  \newcommand{\mol}{\mole}
\else
  \newunit{\cubiccentimeter}{\cm\unitsuperscript{3}}
  \newunit{\Molar}{\textsc{m}}
  \newunit{\molar}{\mole\unitsep\dm\unitsuperscript{--3}}
  \newunit{\mmHg}{mm\unitsep{}Hg}
  \newunit{\mol}{\mole}
\fi
\newcommand{\cmc}{\cubiccentimeter}
\newcommand{\Hz}{\hertz}
```

\cst@varnothing
\standardstate

In a very similar vein, the "standard state" symbol is handy. This is produced by rotating the "varnothing" symbol from the AMS set. Note that the rotation angle here has been carefully checked, but is set by eye. The symbol is loaded directly here, rather than using the `amssymb` package, to avoid any clashes.

```
\DeclareSymbolFont{CSTAMS}{U}{msb}{m}{n}
\DeclareMathSymbol{\cst@varnothing}{\mathord}{CSTAMS}{"3F}
\providecommand*{\standardstate}%
  {\textsuperscript{\rotatebox[origin=c]{138.8}%
  {\ensuremath{\cst@varnothing}}}}
```

7.5 Float formatting

The next step is to format the floats correctly. Unfortunately, `memoir` does not provide all of the commands needed to achieve this. Thus the `float` package is needed; in order to load it, the `\newfloat` command in `memoir` is killed off. The `chemscheme` package does not load `float` when `memoir` is being used, because creating a new float type is catered for directly by `memoir`. Unfortunately, the `memoir` system isn't flexible enough for what is needed by `chemstyle`, so the hard work of `chemscheme` is undone here! For other document classes, `float` will already have been loaded by `chemscheme`.

```
\@ifclassloaded{memoir}
  {\let\newfloat\@undefined
   \RequirePackage{float}}
  {}
```

The standard float types are now restyled to place the captions correctly (for most journals). Normally in chemical documents the author expects the float to be "here" if possible; this is therefore set as the default.

```
\floatstyle{plaintop}
\restylefloat{table}
\floatstyle{plain}
\restylefloat{scheme}
\restylefloat{figure}
\floatplacement{table}{htbp}
\floatplacement{scheme}{htbp}
\floatplacement{figure}{htbp}
```

The contents of floats are centred by default, using the hook from the `chemstyle` package.

```
\floatcontentscentre
```

7.6 Cross-references

The naming for cross-references is sorted out properly using the `varioref` package.

```
\AtBeginDocument{%
  \labelformat{scheme}{\schemename~#1}}
\labelformat{figure}{\figurename~#1}
\labelformat{table}{\tablename~#1}
```

7.7 Latin words

`\latin` `\cst@latin` `\latinemphon` `\latinemphoff`

A series of Latin phrases are provided, with a quick switch to print them in Roman letters if needed. A mechanism is needed to alter the effect of the `\latin` command *only* if the user does not have their own version. This is achieved here, with precautions taken to ensure the user can define their own `\latin` command *after* loading `chemstyle` and still have everything work properly.

```
\let\latinemphon\relax
\let\latinemphoff\relax
\let\cst@latin\relax
\ifcst@phrases
  \newcommand{\latinemphon}{\let\cst@latin\emph}
  \newcommand{\latinemphoff}{\let\cst@latin\relax}
  \AtBeginDocument{%
    \providecommand\latin{\cst@latin}
  }
\fi
\latinemphon
```

In ACS journals, the abbreviations "*e.g.*" and "*i.e.*" are always followed by a comma. In order to provide an automated way to do this, a series of macros are needed to test the punctuation trailing the macro names. Firstly, a switch is provided for the journal style files.

```
\newif  \ifcst@comma    \cst@commafalse
```

The implementation of the testing code is delayed until the beginning of the document to allow the switching to occur. First the case where a comma is needed is handled.

```
\AtBeginDocument{%
  \ifcst@comma
```

\cst@punct The \cst@punct macro holds the comma-containing punctuation to be added.

```
\def\cst@punct{.,\cst@xspace}
```

\cst@addpunct
\cst@add@punct
\cst@add@punct@ The following macros are very closely based on those in the cite package used for moving citations after punctuation. The first macro is used as an initial hook. Notice that \relax is essential here, as it provides an argument for \cst@add@punct in the first round of checking.

```
\def\cst@addpunct{%
  \cst@add@punct\relax%
}
```

Here, a plain TeX \futurelet is used to test the next character. Notice that this macro takes a single argument, which is used to recursively gobble up punctuation.

```
\def\cst@add@punct#1{%
  \futurelet\@tempa\cst@add@punct@%
}
```

The checking occurs here. If a match is made, then the process is repeated to allow the punctuation to be gobbled.

```
\def\cst@add@punct@{%
  \ifx\@tempa.%
    \let\@tempb\cst@add@punct%
  \else
    \ifx\@tempa,%
      \let\@tempb\cst@add@punct%
```

No match, and so new punctuation is to be added and the loop ended.

```
    \else
      \let\@tempb\cst@punct%
    \fi
  \fi
  \@tempb%
}
\fi
}
```

\cst@xspace The use of xspace is optional; this is handled using a package option and the internal macro \cst@xspace.

```
\ifcst@xspace
  \RequirePackage{xspace}
  \let\cst@xspace\xspace
\else
  \let\cst@xspace\relax
\fi
```

\etc
\invacuo
\etal
\eg
\ie For the macros themselves, care is taken about trailing full stops. The \xspace command deals with any problems of spacing. Things could go wrong with complex punctuation, as no other checks are performed. All of these functions use \providecommand to avoid standing on the user's own versions, if they exist.

```
\ifcst@phrases
  \AtBeginDocument{
    \providecommand{\etc}%
```

```
      {\@ifnextchar.{\cst@etc}{\cst@etc.\cst@xspace}}
    \providecommand{\invacuo}%
      {\latin{in vacuo}\cst@xspace}
    \providecommand{\etal}%
      {\@ifnextchar.{\cst@etal}{\cst@etal.\cst@xspace}}
    \ifcst@comma
      \providecommand{\eg}%
        {\cst@eg\cst@addpunct}
      \providecommand{\ie}%
        {\cst@ie\cst@addpunct}
    \else
      \providecommand{\eg}%
        {\@ifnextchar.{\cst@eg}{\cst@eg.\cst@xspace}}
      \providecommand{\ie}%
        {\@ifnextchar.{\cst@ie}{\cst@ie.\cst@xspace}}
    \fi
  }
\fi
```

`\cst@etal` `\cst@etc` `\cst@eg` `\cst@ie`

Internal macros are used for items ending in a full stop, to allow clean handling of spacing. Notice that `\ie` and `\eg` cannot come at the end of a sentence, they are designed to give only an inter-word space.

```
\def\cst@etal{\latin{et~al}}
\def\cst@etc{\latin{etc}}
\def\cst@ie{\latin{i.e\spacefactor999\relax}}
\def\cst@eg{\latin{e.g\spacefactor999\relax}}
```

7.8 Handling section naming

First, a new if is needed to differentiate between "Notes and References" and "References and Notes."

```
\newif  \ifcst@notesbefore  \cst@notesbeforetrue
```

`\cst@name@refs`

To keep life simple in the main macro, and to save on redundant code, the ordering of "Notes," "and" and "References" is handled here.

```
\def\cst@name@refs#1#2#3{%
  \ifcst@notesbefore
    #3\space#2\space#1%
  \else
    #1\space#2\space#3%
  \fi
}
```

`\cst@language`

The default language of the document is probably English. However, this is alterable if needed, as it is not set in stone. Anyone needing to mess with this is probably happy using `\makeatletter`, so the command is kept out of user space.

```
\def\cst@language{english}
```

`\cst@refsection`

The existence of a suitable reference section name command is checked here. Some document classes (*e.g.* minimal) may not define a suitable command.

Assuming that is not the case, `\cst@refsection` is used to store the name of the macro holding the references section.

```
\@ifundefined{refname}{%
  \@ifundefined{bibname}{%
    \PackageWarning{chemstyle}%
      {No bibliography name command defined in document class}%
    \def\cst@namerefs#1#2#3#4{}%
  }{%
    \def\cst@refsection{\bibname}%
  }
}{%
   \def\cst@refsection{\refname}%
}
```

`\cst@namerefs` `\cst@language@check`

The main macro of the heading-altering section takes four arguments, {⟨*language*⟩}, {⟨*references*⟩}, {⟨*and*⟩} and {⟨*notes*⟩}. The first is the `babel` label for the language that the words are four, and the other three parameters and the words "References", "and" and "Notes", respectively.

```
\@ifundefined{cst@namerefs}{%
  \def\cst@namerefs#1#2#3#4{%
    \ifcst@notes%
```

The non-`babel` version of the function only does anything if the language passed in `#1` is the default document language (probably English). The check for this needs the macro `\cst@language@check` to get `\ifx` to behave as desired.

```
      \def\cst@language@check{#1}
      \ifx\cst@language\cst@language@check
        \expandafter\renewcommand\expandafter{\cst@refsection}%
          {\cst@name@refs{#2}{#3}{#4}}%
      \fi
```

To ensure that things work whether `babel` is loaded or not, a few hoops have to be jumped through. Altering the `babel` strings is delayed until the start of the document, in case `babel` is loaded after `chemstyle`. There is also a complication that MikTeX defines `\languagename` even if `babel` is not loaded. This may contain an undesired value, and so the presence of `babel` needs to be tested for to avoid strange errors.

```
      \AtBeginDocument{%
        \@ifpackageloaded{babel}{%
          \expandafter\addto\expandafter{\csname captions#1\endcsname}%
            {\expandafter\renewcommand\expandafter{\cst@refsection}%
            {\cst@name@refs{#2}{#3}{#4}}}%
          \expandafter\selectlanguage\expandafter{\languagename}}
        {}}
    \else%
      \ifx#1\cst@language
        \expandafter\renewcommand\expandafter{\cst@refsection}{#2}%
      \fi
      \AtBeginDocument{%
        \@ifpackageloaded{babel}{%
          \expandafter\addto\expandafter{\csname captions#1\endcsname}%
            {\expandafter\renewcommand\expandafter{\cst@refsection}{#2}}%
          \expandafter\selectlanguage\expandafter{\languagename}}%
        {}}
```

```
    \fi%
  }
}{}
```

Finally, default names are loaded for a range of languages. Most journals stick to the same words, with only the order changing.

```
\cst@namerefs{english}{References}{and}{Notes}
\cst@namerefs{UKenglish}{References}{and}{Notes}
\cst@namerefs{ngerman}{Literatur}{und}{Notizen}
\cst@namerefs{german}{Literatur}{und}{Notizen}
\cst@namerefs{french}{R\'ef\'erences}{et}{Notes}
\cst@namerefs{frenchb}{R\'ef\'erences}{et}{Notes}
```

7.9 Loading the style definition

The style definition is loaded here, once else everything is in place. A style must be loaded, so a default is provided to be on the safe side. The journal style file must have extension `.jdf`.

```
\ifx\cst@option\relax
  \PackageWarning{chemstyle}
    {You didn't say which style to use \MessageBreak
     Defaulting to the ``no change'' style: none}
  \def\cst@option{none}
\fi
\InputIfFileExists{\cst@option.jdf}
  {\PackageInfo{chemstyle}
    {Loaded \cst@option.jdf}}
  {\PackageWarning{chemstyle}
    {Required style \cst@option\space does not exist}}
```

8 Configuration files

Each journal style needs slightly differing commands to get the formatting just right. This is handled here, with each style in a separate file. There is not a lot happening in most of these files, as the information is by its nature quite repetitive.

8.1 RSC style

```
\ProvidesFile{rsc.jdf}[2007/10/31 v1.1h]
\AtBeginDocument{%
  \renewcommand{\figurename}{Fig.}}
\captionsetup{labelsep=quad,labelfont=bf}
\let\cst@emph\emph
\def\cst@hyph{-}
\let\cst@super\relax
\cst@prefixtrue
\latinemphon
\cst@commafalse
```

8.2 *Angew. Chem.* style

```
\ProvidesFile{angew.def}[2007/10/31 v1.1h]
\captionsetup{labelsep=period,labelfont={bf,it},font=sf,singlelinecheck=off}
\captionsetup[table]{labelsep=colon}
\let\cst@emph\emph
\def\cst@hyph{}
\let\cst@super\relax
\cst@prefixtrue
\latinemphoff
\cst@commafalse
```

8.3 *J. Organomet. Chem.* style

```
\ProvidesFile{jomc.def}[2007/10/31 v1.1h]
\captionsetup{labelsep=period}
\captionsetup[table]{labelsep=newline,singlelinecheck=off}
\AtBeginDocument{%
  \renewcommand{\figurename}{Fig.}}
\let\cst@emph\emph
\def\cst@hyph{}
\let\cst@super\textsuperscript
\cst@prefixtrue
\latinemphoff
\cst@commafalse
```

8.4 *Tetrahedron Lett.* style

```
\ProvidesFile{tetlett.def}[2007/10/31 v1.1h]
\captionsetup{labelsep=period,singlelinecheck=off,labelfont=bf}
\let\cst@emph\emph
\def\cst@hyph{-}
\let\cst@super\relax
\cst@prefixtrue
\latinemphoff
\cst@commatrue
```

8.5 *J. Am. Chem. Soc.* style

```
\ProvidesFile{jacs.jdf}[2007/10/31 v1.1h]
\DeclareCaptionLabelSeparator{perquad}{.\quad}
\captionsetup{labelfont={bf,it,sf},textfont=sf,labelsep=perquad}
\captionsetup[figure]{textfont=rm}
\captionsetup{singlelinecheck=off}
\let\cst@emph\emph
\def\cst@hyph{}
\let\cst@super\textsuperscript
\cst@prefixtrue
\floatstyle{plaintop}
\restylefloat{scheme}
\floatstyle{plain}
\latinemphoff
\cst@notesbeforefalse
\cst@commatrue
```

8.6 *Inorg. Chem.* style

Almost exactly the same as for *J. Am. Chem. Soc.*, so most of the work is left to `jacs.jdf`.

```
\ProvidesFile{ic.jdf}[2007/10/31 v1.1h]
\input {jacs.jdf}
\captionsetup{textfont=rm}
```

8.7 *J. Phys. Chem.* style

```
\ProvidesFile{jpc.jdf}[2007/10/31 v1.1h]
\DeclareCaptionFormat{labelcaps}{\MakeUppercase{#1}#2#3}
\captionsetup{font=bf,labelsep=colon,format=labelcaps}
\captionsetup[figure]{format=plain,textfont=md,labelsep=period}
\captionsetup{singlelinecheck=off}
\let\cst@emph\emph
\def\cst@hyph{}
\let\cst@super\textsuperscript
\cst@prefixtrue
\floatstyle{plaintop}
\restylefloat{scheme}
\floatstyle{plain}
\latinemphoff
\cst@notesbeforefalse
\cst@commatrue
```

8.8 *Org. Lett.* style

```
\ProvidesFile{orglett.jdf}[2007/10/31 v1.1h]
\RequirePackage{xcolor}
```

`\OrgLettColour` To allow the user to control the colour of the "bars" in this style, an additional macro is provided. The `purple` colour looks about right for matching the printed journal, but if anyone has a better suggestion please let the package author know.

```
\newcommand*{\OrgLettColour}{purple}
```

In order to get the distinctive coloured bars used by *Org. Lett.*, a new style for floats is needed. This is based on the `ruled` style from the `float` package.

```
\newcommand\fs@orglett{\def\@fs@cfont{\bfseries}
  \let\@fs@capt\floatc@ruled
  \def\@fs@pre{\begingroup\color{\OrgLettColour}
    \hrule height12pt depth0pt \kern2pt\endgroup}%
```

This is the same `\@fs@mid` as is used in the `float` package for `plaintop` floats.

```
  \def\@fs@mid{\vspace\belowcaptionskip\relax}%
  \def\@fs@post{\begingroup\color{\OrgLettColour}\kern2pt
    \hrule height1.5pt depth0pt\endgroup}%
  \let\@fs@iftopcapt\iftrue}
```

Figures need to be slightly different, so an almost identical command is needed.

```
\newcommand\fs@orglettfig{\def\@fs@cfont{\bfseries}
  \let\@fs@capt\floatc@plain
  \def\@fs@pre{\begingroup\color{\OrgLettColour}
    \hrule height12pt depth0pt \kern2pt\endgroup}%
  \def\@fs@mid{\vspace\abovecaptionskip\relax}%
  \def\@fs@post{\begingroup\color{\OrgLettColour}\kern2pt
    \hrule height1.5pt depth0pt\endgroup}%
  \let\@fs@iftopcapt\iffalse}
```

The new style is now applied. Users can change back to normal floats by changing back to the `plain` style.

```
\floatstyle{orglettfig}
\restylefloat{figure}
\floatstyle{orglett}
\restylefloat{scheme}
\restylefloat{table}
```

The more usual style commands now occur.

```
\DeclareCaptionLabelSeparator{perquad}{.\quad}
\captionsetup{labelfont=bf,labelsep=perquad}
\let\cst@emph\emph
\def\cst@hyph{}
\let\cst@super\textsuperscript
\cst@prefixtrue
\latinemphoff
\cst@notesbeforefalse
\cst@commatrue
```

8.9 The empty style — none

To allow the user to load the extra macros provided here without any style changes, a "do nothing" style is provided. It simply makes sure that very little changes compared to the LaTeX kernel. This requires undoing the defaults provided above. For commands where a default is needed (*e.g.* the `\latin` command) the style of the RSC is followed. As the `float` package has been loaded, notice that captions will be placed below floats even if the `\caption` command appears above the contents of the floating environment.

```
\ProvidesFile{none.jdf}[2007/10/31 v1.1h]
\floatstyle{plain}
\restylefloat{table}
\labelformat{scheme}{#1}
\labelformat{figure}{#1}
\labelformat{table}{#1}
\floatplacement{table}{tbp}
\floatplacement{scheme}{tbp}
\floatplacement{figure}{tbp}
```

9 Change History

10 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.