

KOMA-Script File `scrhack.dtx` *

Markus Kohm[†]

package

Some packages from other authors may have problems with KOMA-Script. In my opinion some packages could be improved. With some packages this makes only sense, if KOMA-Script was used. With some other packages the package author has another opinion. Sometimes proposals was never answered. Package `scrhack` contains all those improvement proposals for other packages. This means, `scrhack` redefines macros of packages from other authors! The redefinitions are only activated, if those packages were loaded. Users may prevent `scrhack` from redefining macros of individual packages.

Contents

1	The hyperref hack	2
2	The float hack	2
3	The floatrow hack	3
4	The listings hack	4
5	The setspace hack	4
6	The lscape hack	5
7	Implementation of <code>scrhack</code>	5
7.1	Optionen	5
7.2	Verwendete Anweisungen	6
7.3	Der hyperref-Hack	7

*This file is version (hacking of `scrhack.dtx`.

[†][mailto:komascript\(at\)gmh.info](mailto:komascript(at)gmh.info)

7.4	Der float-Hack	10
7.5	Der floatrow-Hack	13
7.6	Der listings-Hack	15
7.7	Der setspace-Hack	17
7.8	Der lscape-Hack	20
7.9	Optionen ausführen	21

1 The hyperref hack

Before version 6.79h package `hyperref` does behave different at part, chapter, and section headings that get no number. If they get no number, because of to low counter

`secnumdepth` `hyperref` sets an anchor for links and bookmarks before the heading. Same would be, if the headings have a number. But if the headings get no number because of usage of the star version of the commands, e.g., `\part*`, `\chapter*` or `\section*`, the anchor for links and bookmarks are set after the headings. The anchors for numbered headings are always set before the headings.

Package `scrhack` redefines some macros of some `hyperref` driver files, e.g., `hpdftex.def`, after loading the `hyperref` driver file. With this redefinitions the anchor of not numbered headings will be set always before the headings, too.

You may switch off the `hyperref` hack loading package `scrhack` with option `hyperref=false`. You may also switch off the `hyperref` hack using `\KOMAOptions{hyperref=false}` or `\KOMAoption{hyperref}{false}` somewhere after loading package `scrhack`, but before loading the `hyperref` driver package, that is by default after loading the package.

2 The float hack

Package `float` uses macros `\float@listhead` to set the headings of a float listing and `\float@addtolists` to add informations to all float listings. These macros where proposed by the `KOMA-Script` author for some years. In theory those macros may be used by several class and package authors to deligate some parts of the creation of a float listing to the class. This would increase the compatibility of packages and classes. But unfortunately some package authors, even the author of package `float`, implemented the commands in such a way, that these packages will become incompatible to each other.

Because of this KOMA-Script stopped support for `\float@addtolists` and `\float@listhead` with version 3. Instead of this KOMA-Script supports several improvements for package authors using KOMA-Script package `tocbasic`.

Package `scrhack` redefines some macros of package `float` to not longer use `\float@addtolists` and `\float@listhead` but use the interface of package `tocbasic`. This does not only improve the compatibility of KOMA-Script and package `float`, but also improves the compatibility of packages `babel` and `float`.

You may switch off the float hack loading package `scrhack` with option `float=false`. You may also switch off the float hack using `\KOMAoptions{float=false}` or `\KOMAoption{float}{false}` somewhere after loading package `scrhack`, but before loading package `float`.

3 The floatrow hack

Package `floatrow` uses macros `\float@listhead` to set the headings of a float listing and `\float@addtolists` to add informations to all float listings. These macros were proposed by the KOMA-Script author for some years. In theory those macros may be used by several class and package authors to delegate some parts of the creation of a float listing to the class. This would increase the compatibility of packages and classes. But unfortunately some package authors, even the author of package `floatrow`, implemented the commands in such a way, that these packages will become incompatible to each other.

Because of this KOMA-Script stopped support for `\float@addtolists` and `\float@listhead` with version 3. Instead of this KOMA-Script supports several improvements for package authors using KOMA-Script package `tocbasic`.

Package `scrhack` redefines some macros of package `floatrow` to not longer use `\float@addtolists` and `\float@listhead` but use the interface of package `tocbasic`. This does not only improve the compatibility of KOMA-Script and package `floatrow`, but also improves the compatibility of packages `babel` and `floatrow`.

You may switch off the floatrow hack loading package `scrhack` with option `floatrow=false`. You may also switch off the floatrow hack using `\KOMAoptions{floatrow=false}` or `\KOMAoption{floatrow}{false}` somewhere after loading package `scrhack`, but before loading package `floatrow`.

4 The listings hack

Package `listings` uses macros `\float@listhead` to set the headings of a float listing, if defined, and `\float@addtolists` to add informations to all float listings. These macros were proposed by the KOMA-Script author for some years. In theory those macros may be used by several class and package authors to delegate some parts of the creation of a float listing to the class. This would increase the compatibility of packages and classes. But unfortunately some package authors, even the author of package `float`, implemented the commands in such a way, that these packages may become incompatible to each other.

Because of this KOMA-Script stopped support for `\float@addtolists` and `\float@listhead` with version 3. Instead of this KOMA-Script supports several improvements for package authors using KOMA-Script package `tocbasic`.

Package `scrhack` redefines some macros of package `listings` to not longer use `\float@addtolists` and `\float@listhead` but use the interface of package `tocbasic`. This does not only improve the compatibility of KOMA-Script and package `listings`, but also improves the compatibility of packages `babel` and `listings`.

Note: A significant change with `scrhack` is, that KOMA-Script options like `lists=totoc` or `lists=totocnumbered` does only change the behaviour of `\listoflistings`, if they are set after loading package `listings`.

You may switch off the listings hack loading package `scrhack` with option `listings=false`. You may also switch off the listings hack using `\KOMAoptions{listings=false}` or `\KOMAoption{listings}{false}` somewhere after loading package `scrhack`, but before loading package `listings`.

5 The setspace hack

Package `setspace` defines macros `\onehalfspacing` and `\doublespacing` using `\@ptsize` as an argument of `\ifcase`. But if `\@ptsize` is not an integer but a real number, this fails, because the digits from the decimal points are interpreted as text of that case. Several solutions for this are thinkable. I've decided to redefine `\onehalfspacing` and `\doublespacing`. The new definition is more general and somehow more exact.

You can switch of the `setspace` hack loading package `scrhack` with option `setspace=false`. You may also switch of the `setspace` hack using `\KOMAOptions{setspace=false}` or `\KOMAoption{setspace}{false}` somewhere after loading package `scrhack`, but before loading package `setspace`.

Note: If you want to use `setspace` with package option `onehalfspacing` or `doublespacing` you have to load `scrhack` before `setspace`.

6 The `lscape` hack

Package `lscape` defines an environment `landscape` to set the page contents but not head and foot landscape. Inside this environment it changes `\textheight` to the value of `\textwidth`, but it does not change `\textwidth` to the former value of `\textheight`. This is inconsistent. As far a I know, `\textwidth` is unchanged because setting it to `\textheight` could blame other packages or user commands. But changing `\textheight` could also blame other packages or user commands and indeed it breaks, e. g., `showframe` and `scrlayer`. So best would be, not to change `\textheight`, too. `scrhack` uses package `xpatch` to modify the environment start macro `\landscape` appropriately.

You can switch of the `lscape` hack loading package `scrhack` with option `lscape=false`. You can also change option `lscape` afterwards. If the option is `false` while loading `lscape`, `scrhack` will not patch `\landscape` and later changes of the option have no effect. But if the option is `true` while loading `lscape` or if `scrhack` is loaded after `lscape` without option `lscape=false`, every later change of the option using `\KOMAoption` or `\KOMAOptions` will have the expected effect.

7 Implementation of `scrhack`

7.1 Optionen

Das Paket bedient sich `\KOMAOptions` etc. aus `scrkbase` (dieses wird übrigens direkt per `scrkbase.dtx` geladen).

Per Option kann gewählt werden, welche Manipulationen geladen werden sollen. Alle diese Optionen können jedoch nur bis zum Laden des entsprechenden Pakets oder dem Laden von `scrhack` gesetzt werden (es zählt, was später kommt). Anschließend sind sie wirkungslos.

7.2 Verwendete Anweisungen

`\scr@ifexpected` Wenn die im ersten Argument angegebene Anweisung nach Ausführung der im zweiten Argument angegebenen Anweisungen unverändert ist, dann soll das dritte Argument ausgeführt werden, sonst das vierte.

```
1 \newcommand{\scr@ifexpected}[2]{%
2   \begingroup
3     \let\@tempa#1
4     #2
5     \ifx\@tempa#1
6       \aftergroup\@firstoftwo
7     \else
8       \aftergroup\@secondoftwo
9     \fi
10  \endgroup
11 }
```

`\scr@hack@load` Wenn die Datei mit dem Namen des zweiten Arguments und der Endung des ersten Arguments so geladen wurde, dass L^AT_EX eine Versionsinfo dazu gespeichert hat, dann soll zusätzlich der entsprechende Hack geladen werden.

```
12 \newcommand*{\scr@hack@load}[2]{%
13   \expandafter\ifx\csname ver@#2.#1\endcsname\relax
14     \expandafter\@secondoftwo
15   \else
```

Allerdings wird jeder Hack nur genau einmal geladen:

```
16     \expandafter\ifx\csname ver@#2.hak\endcsname\relax
17       \expandafter\expandafter\expandafter\@firstoftwo
18     \else
19       \expandafter\expandafter\expandafter\@secondoftwo
20     \fi
21   \fi
22   {%
23     \PackageInfo{scrhack}{loading #2 hack}%
24     \edef\reserved@a{%
25       \noexpand\makeatletter\noexpand\input{#2.hak}%
26       \noexpand\catcode'\noexpand\@the\catcode'\@relax
27     }\reserved@a
28   }{%
29     \PackageInfo{scrhack}{ignoring #2 hack}%
30   }%
31 }
```

7.3 Der hyperref-Hack

hyperref setzt den Anker zu der Stern-Variante einer Überschrift hinter die Überschrift, während es bei der nicht Stern-Variante den Anker auch dann vor die Überschrift setzt, wenn die Überschrift aufgrund von `secnumdepth` nicht nummeriert wird. Der Hack setzt den Anker einheitlich vor die Überschrift.

hyperref

```
32 <*package & option>
33 \KOMA@ifkey{hyperref}{@scrhack@hyperref}%
34 \KOMAExecuteOptions{hyperref=true}%
35 </package & option>
36 <*package & body>
```

Hier muss ein wenig trickreicher gearbeitet werden, weil hyperref die Treiberdatei per `\AtEndOfPackage` lädt und der Hack erst danach installiert werden darf. Mit `\AfterPackage*` alleine, würde der Hack aber vor dem Laden der Treiberdatei installiert. Dafür können wir aber sicher sein, dass ein innerhalb von `\AfterPackage*` aufgerufenes `\AtEndOfPackage` garantiert nach dem Laden der Treiberdatei ausgeführt wird. Das funktioniert auch noch, wenn hyperref bereits geladen wurde. In dem Fall wird der Code einfach nach dem Ende von `scrhack` statt nach dem Ende von `hyperref` ausgeführt.

```
37 \BeforePackage{hyperref}{%
38   \scr@ifundefinedorrelax{hy@insteadofrefstepcounter}{}{%
39     \PackageInfo{scrhack}{hyperref hack deactivated because of\MessageBreak
40       detection of KOMA-Script class, that doesn't\MessageBreak
41       need that hack,}%
42     \KOMAExecuteOptions[.scrhack.sty]{hyperref=false}%
43   }%
44 }
45 \AfterPackage*{hyperref}{%
46   \if@scrhack@hyperref
47     \@ifpackagelater{hyperref}{2009/11/24}{%
48       \PackageInfo{scrhack}{hyperref hack deactivated because of\MessageBreak
49         detection of hyperref version, that doesn't\MessageBreak
50         need that hack,}%
51       \KOMAExecuteOptions[.scrhack.sty]{hyperref=false}%
52     }{%
53       \AtEndOfPackage{%
54         \KOMA@key[.scrhack.sty]{hyperref}{%
55           \PackageWarning{scrhack}{option 'hyperref=#1' ignored}%
56           \FamilyKeyStateProcessed
57         }%
58       }
59     }
60   }%
61 }
```

```

58         \if@scrhack@hyperref\scr@hack@load\@pkgextension{hyperref}\fi
59     }%
60 }%
61 \fi
62 }
63 </package & body>

```

`\@schapter` Eigentlich wird hier gar nicht `hyperref.sty` verändert, sondern diverse
`\@spart` Treiberdateien. Sobald das Paket `hyperref` geladen ist, ist auch die passende
`\@ssect` Treiberdatei geladen und außerdem sind alle Treiberdateien, die entsprechende
Definitionen vornehmen, gleichermaßen betroffen. Also kann der entsprechende
Patch einfach erfolgen, wenn `hyperref` geladen ist (was bereits von `\scr@hack@load`
getestet wurde). Es muss also nur noch sichergestellt werden, dass die
umzudefinierenden Macros derzeit den erwarteten Inhalt haben.

```

64 <*hyperref & body>
65 \scr@ifexpected\@schapter{%
66     \def\@schapter#1{%
67         \H@old@schapter{#1}%
68         \begingroup
69             \let\@mkboth\@gobbletwo
70             \Hy@GlobalStepCount\Hy@linkcounter
71             \xdef\@currentHref{\Hy@chapapp*.\the\Hy@linkcounter}%
72             \Hy@raisedlink{%
73                 \hyper@anchorstart{\@currentHref}\hyper@anchorend
74             }%
75         \endgroup
76     }%
77 }{%
78     \PackageInfo{scrhack}{redefining \string\@schapter}%
79     \def\@schapter#1{%
80         \begingroup
81             \let\@mkboth\@gobbletwo
82             \Hy@GlobalStepCount\Hy@linkcounter
83             \xdef\@currentHref{\Hy@chapapp*.\the\Hy@linkcounter}%
84             \Hy@raisedlink{%
85                 \hyper@anchorstart{\@currentHref}\hyper@anchorend
86             }%
87         \endgroup
88         \H@old@schapter{#1}%
89     }%
90 }{%
91     \scr@ifexpected\@schapter{%
92         \def\@schapter#1{%
93             \begingroup
94                 \let\@mkboth\@gobbletwo
95                 \Hy@GlobalStepCount\Hy@linkcounter
96                 \xdef\@currentHref{\Hy@chapapp*.\the\Hy@linkcounter}%

```



```

97         \Hy@raisedlink{%
98         \hyper@anchorstart{\@currentHref}\hyper@anchorend
99         }%
100     \endgroup
101     \H@old@schapter{#1}%
102     }%
103 }{}{%
104     \PackageWarningNoLine{scrhack}{unknown \string\@schapter\space
105     definition found!\MessageBreak
106     Maybe you are using a unsupported hyperref version}%
107 }%
108 }
109
110 \scr@ifexpected\@spart{%
111     \def\@spart#1{%
112         \H@old@spart{#1}%
113         \Hy@GlobalStepCount\Hy@linkcounter
114         \xdef\@currentHref{part*.\the\Hy@linkcounter}%
115         \Hy@raisedlink{%
116         \hyper@anchorstart{\@currentHref}\hyper@anchorend
117         }%
118     }%
119 }{%
120     \PackageInfo{scrhack}{redefining \string\@spart}%
121     \def\@spart#1{%
122         \Hy@GlobalStepCount\Hy@linkcounter
123         \xdef\@currentHref{part*.\the\Hy@linkcounter}%
124         \Hy@raisedlink{%
125         \hyper@anchorstart{\@currentHref}\hyper@anchorend
126         }%
127         \H@old@spart{#1}%
128     }%
129 }{%
130     \scr@ifexpected\@spart{%
131         \def\@spart#1{%
132             \Hy@GlobalStepCount\Hy@linkcounter
133             \xdef\@currentHref{part*.\the\Hy@linkcounter}%
134             \Hy@raisedlink{%
135             \hyper@anchorstart{\@currentHref}\hyper@anchorend
136             }%
137             \H@old@spart{#1}%
138         }%
139     }{}{%
140         \PackageWarningNoLine{scrhack}{unknown \string\@spart\space
141         definition found!\MessageBreak
142         Maybe you are using a unsupported hyperref version}%
143     }%
144 }
145

```

```

146 \scr@ifexpected\@ssect{%
147   \def\@ssect#1#2#3#4#5{%
148     \H@old@ssect{#1}{#2}{#3}{#4}{#5}%
149     \phantomsection
150   }%
151 }{%
152   \PackageInfo{scrhack}{redefining \string\@ssect}%
153   \def\@ssect#1#2#3#4#5{%
154     \H@old@ssect{#1}{#2}{#3}{#4}{\phantomsection\ignorespaces#5}%
155   }%
156 }{%
157   \scr@ifexpected\@ssect{%
158     \def\@ssect#1#2#3#4#5{%
159       \H@old@ssect{#1}{#2}{#3}{#4}{\phantomsection\ignorespaces#5}%
160     }%
161   }{}{%
162     \PackageWarningNoLine{scrhack}{unknown \string\@ssect\space
163       definition found!\MessageBreak
164       Maybe you are using a unsupported hyperref version}%
165   }%
166 }
167 </hyperref & body>

```

7.4 Der float-Hack

Das float-Paket verwendet das Makro `\float@listhead` zum Setzen der Überschriften. Dies wird seit KOMA-Script 3 nicht mehr empfohlen und fliegt demnächst komplett aus der Unterstützung. Stattdessen wird empfohlen, dass Pakete `tocbasic` unterstützen. Der Aufwand dafür ist sehr gering und wird mit vielen neuen Möglichkeiten belohnt.

Dieser Hack rüstet die `tocbasic`-Unterstützung für `float` nach.

`float`

```

168 <*package & option>
169 \KOMA@ifkey{float}{\@scrhack@float}%
170 \KOMAExecuteOptions{float=true}%
171 </package & option>
172 <*package & body>
173 \AfterPackage*{float}{%
174   \KOMA@key[.scrhack.sty]{float}{%
175     \PackageWarning{scrhack}{option ‘float’ ignored}%
176     \FamilyKeyStateProcessed
177   }%
178   \if@scrhack@float\scr@hack@load\@pkgextension{float}\fi
179 }
180 </package & body>

```

`\newfloat` Über die Anweisung `\newfloat` wird eine neue Gleitumgebung definiert. Hier muss die neue Erweiterung aus dem dritten Argument `tocbasic` bekannt gemacht werden.

`\listof` Über die Anweisung `\listof` wird ein Verzeichnis für Gleitumgebungen ausgegeben. Hier muss schlicht die entsprechende Anweisung von `tocbasic` verwendet werden.

`\float@addtolists` Diese Anweisung wird nicht länger benötigt und daher auf die ursprüngliche Definition zurückgesetzt.

```

181 <*float & body>
182 \scr@ifexpected{\newfloat}{%
183   \long\def\newfloat#1#2#3{\@namedef{ext@#1}{#3}
184     \let\float@do=\relax
185     \xdef\@tempa{\noexpand\float@exts{\the\float@exts \float@do{#3}}}%
186     \@tempa
187     \floatplacement{#1}{#2}%
188     \@ifundefined{fname@#1}{\floatname{#1}{#1}}{}
189     \expandafter\edef\csname ftype@#1\endcsname{\value{float@type}}%
190     \addtocounter{float@type}{\value{float@type}}
191     \restylefloat{#1}%
192     \expandafter\edef\csname fnum@#1\endcsname%
193     {\expandafter\noexpand\csname fname@#1\endcsname{}
194       \expandafter\noexpand\csname the#1\endcsname}
195     \@ifnextchar[%]
196     {\float@newx{#1}}%
197     {\@ifundefined{c@#1}{\newcounter{#1}\@namedef{the#1}{\arabic{#1}}}%
198       {}}}%
199 }{%
200   \scr@ifexpected{\listof}{%
201     \def\listof#1#2{%
202       \@ifundefined{ext@#1}{\float@error{#1}}{%
203         \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
204         \float@listhead{#2}%
205         \begingroup\setlength{\parskip}{\z@}%
206         \@starttoc{\@nameuse{ext@#1}}%
207         \endgroup}}%
208   }{%
209     \RequirePackage{tocbasic}%
210     \PackageInfo{scrhack}{redefining \string\newfloat}%
211     \renewcommand\newfloat[3]{%
212       \ifattoclist{#3}{%
213         \PackageError{scrhack}{extension ‘#3’ already in use}{%
214           Each extension may be used only once.\MessageBreak
215           You, the class, or another package already uses extension
216           ‘#3’.\MessageBreak
217         \string\newfloat\space command will be ignored!}%

```

```

218 }{%
219   \addtotoclist[float]{#3}%
220   \setuptoc{#3}{chapteratlist}%
221   \@namedef{ext@#1}{#3}%
222   \let\float@do=\relax
223   \xdef\@tempa{\noexpand\float@exts{\the\float@exts \float@do{#3}}}%
224   \@tempa
225   \floatplacement{#1}{#2}%
226   \@ifundefined{fname@#1}{\floatname{#1}{#1}}{}%
227   \expandafter\edef\csname ftype@#1\endcsname{\value{float@type}}%
228   \addtocounter{float@type}{\value{float@type}}
229   \restylefloat{#1}%
230   \expandafter\edef\csname fnum@#1\endcsname%
231   {\expandafter\noexpand\csname fname@#1\endcsname{
232     \expandafter\noexpand\csname the#1\endcsname}%
233   \@ifnextchar[%]
234   {\float@newx{#1}}%
235   {\@ifundefined{c@#1}{\newcounter{#1}\@namedef{the#1}{\arabic{#1}}}%
236     {}}%
237 }%
238 \PackageInfo{scrhack}{redefining \string\listof}%
239 \renewcommand*\listof[2]{%
240   \@ifundefined{ext@#1}{\float@error{#1}}{%
241     \@ifundefined{l@#1}{\expandafter\let\csname l@#1\endcsname\l@figure
242       \@ifundefined{l@#1}{%
243         \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}}%
244     }{}%
245     \listoftoc[{#2}]{\csname ext@#1\endcsname}%
246   }%
247 }%
248 \scr@ifexpected{\float@addtolists}{%
249   \long\def\float@addtolists#1{%
250     \def\float@do##1{\addtocontents{##1}{#1} \the\float@exts}%
251   }{%
252     \PackageInfo{scrhack}{undefining \string\float@addtolists}%
253     \let\float@addtolists\relax
254   }{%
255     \PackageWarningNoLine{scrhack}{unkown \string\float@addtolists\space
256       definition found!\MessageBreak
257       Maybe you are using a unsupported float version}%
258   }%
259 }%
260 \PackageWarningNoLine{scrhack}{unknown \string\listof\space
261   definition found!\MessageBreak
262   Maybe you are using a unsupported float version}%
263 }%
264 }{%
265   \PackageWarningNoLine{scrhack}{unknown \string\newfloat\space
266     definition found!\MessageBreak

```

```

267     Maybe you are using a unsupported float version}%
268 }
269 </float & body>

```

7.5 Der floatrow-Hack

Das floatrow-Paket verwendet das Makro `\float@listhead` zum Setzen der Überschriften. Dies wird seit KOMA-Script 3 nicht mehr empfohlen und fliegt demnächst komplett aus der Unterstützung. Stattdessen wird empfohlen, dass Pakete `tocbasic` unterstützen. Der Aufwand dafür ist sehr gering und wird mit vielen neuen Möglichkeiten belohnt.

Dieser Hack rüstet die `tocbasic`-Unterstützung für floatrow nach.

floatrow

```

270 <*package & option>
271 \KOMA@ifkey{floatrow}{@scrhack@floatrow}
272 \KOMAEecuteOptions{floatrow=true}
273 </package & option>
274 <*package & body>
275 \AfterPackage*{floatrow}{%
276   \KOMA@key[.scrhack.sty]{floatrow}{%
277     \PackageWarning{scrhack}{option 'floatrow' ignored}%
278     \FamilyKeyStateProcessed
279   }%
280   \if@scrhack@floatrow\scr@hack@load\@pkgextension{floatrow}\fi
281 }
282 </package & body>

```

`\DeclareNewFloatType` Über die Anweisung `\DeclareNewFloatType` wird eine neue Gleitumgebung definiert. Hier muss die neue Erweiterung aus dem dritten Argument `tocbasic` bekannt gemacht werden.

`\listof` Über die Anweisung `\listof` wird ein Verzeichnis für Gleitumgebungen ausgegeben. Hier muss schlicht die entsprechende Anweisung von `tocbasic` verwendet werden.

`\float@addtolists` Diese Anweisung wird nicht länger benötigt und daher auf die ursprüngliche Definition zurückgesetzt.

```

283 <*floatrow & body>
284 \scr@ifexpected{\DeclareNewFloatType}{%
285   \long\def\DeclareNewFloatType#1#2{\def\FB@captype{#1}%
286     \expandafter\edef\csname ftype@#1\endcsname{\the\c@float@type}%
287     \addtocounter{float@type}{\value{float@type}}}%

```

```

288 \@namedef{#1name}{#1}\newcounter{#1}%
289 \expandafter\edef\csname fnum@#1\endcsname
290 {\expandafter\noexpand\csname #1name\endcsname\nobreakspace
291 \expandafter\noexpand\csname the#1\endcsname}%
292 \@namedef{the#1}{\arabic{#1}}\flnew@ext{lo#1}\@namedef{fps@#1}{tbp}%
293 \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
294 \caption@setkeys[floatrow]{newfloat}{#2}\let\FR@tmp=\relax
295 \xdef\@tempa{\noexpand\flow@types{\the\flow@types \FR@tmp{#1}}}%
296 \@tempa}%
297 }{%
298 \scr@ifexpected{\listof}{%
299 \def\listof#1#2{%
300 \ifundefined{ext@#1}{\flow@error{Unknown float style ‘#1’}}{%
301 \expandafter\providecommand\csname l@#1\endcsname
302 {\@dottedtocline{1}{1.5em}{2.3em}}%
303 \float@listhead{#2}%
304 \begingroup\setlength{\parskip}{\z@}%
305 \@starttoc{\@nameuse{ext@#1}}%
306 \endgroup}}%
307 }{%
308 \RequirePackage{tocbasic}%
309 \PackageInfo{scrhack}{redefining \string\DeclareNewFloatType}%

```

Eigentlich wäre es besser, wie im float-Hack einen Test vorzuschalten, ob die Dateieindung bereits in Gebrauch ist. Aber das würde voraussetzen, dass die Reihenfolge der Anweisungen geändert wird. Dazu stecke ich aber im Code von floatrow zu wenig drin. (*Note: It would be better to first test, if the new extension is already in use like done at the float hack. But I don't know the floatrow code good enough to make such a change!*)

```

310 \renewcommand\DeclareNewFloatType[2]{\def\FB@captype{#1}%
311 \expandafter\edef\csname ftype@#1\endcsname{\the\c@float@type}%
312 \addtocounter{float@type}{\value{float@type}}%
313 \@namedef{#1name}{#1}\newcounter{#1}%
314 \expandafter\edef\csname fnum@#1\endcsname
315 {\expandafter\noexpand\csname #1name\endcsname\nobreakspace
316 \expandafter\noexpand\csname the#1\endcsname}%
317 \@namedef{the#1}{\arabic{#1}}\flnew@ext{lo#1}\@namedef{fps@#1}{tbp}%
318 \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
319 \caption@setkeys[floatrow]{newfloat}{#2}\let\FR@tmp=\relax
320 \xdef\@tempa{\noexpand\flow@types{\the\flow@types \FR@tmp{#1}}}%
321 \@tempa
322 \xdef\@tempa{\noexpand\addtotoclist[float]{\@nameuse{ext@FB@captype}}%
323 \noexpand\setuptoc{\@nameuse{ext@FB@captype}}{chapteratlist}%
324 }%
325 \@tempa
326 }%
327 \PackageInfo{scrhack}{redefining \string\listof}%
328 \renewcommand*\listof[2]{%

```

```

329 \ifundefined{ext@#1}{\flow@error{Unknown float style '#1'}}{%
330 \ifundefined{l@#1}{\expandafter\let\csname l@#1\endcsname\l@figure
331 \ifundefined{l@#1}{%
332 \namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}}{%
333 }{%
334 \listoftoc[{#2}]{\csname ext@#1\endcsname}%
335 }%
336 }%
337 \scr@ifexpected{\float@addtolists}{%
338 \long\def\float@addtolists#1{%
339 \def\float@do##1{\addtocontents{##1}{#1}} \the\float@exts}%
340 }{%
341 \PackageInfo{scrhack}{undefining \string\float@addtolists}%
342 \let\float@addtolists\relax
343 }{%
344 \PackageWarningNoLine{scrhack}{unkown \string\float@addtolists\space
345 definition found!\MessageBreak
346 Maybe you are using a unsupported floatrow version}%
347 }%
348 }{%
349 \PackageWarningNoLine{scrhack}{unknown \string\listof\space
350 definition found!\MessageBreak
351 Maybe you are using a unsupported floatrow version}%
352 }%
353 }{%
354 \PackageWarningNoLine{scrhack}{unknown \string\DeclareNewFloatType\space
355 definition found!\MessageBreak
356 Maybe you are using a unsupported floatrow version}%
357 }
358 </floatrow & body>

```

7.6 Der listings-Hack

Das listings-Paket verwendet das Makro `\float@listhead` zum Setzen der Überschriften. Dies wird seit KOMA-Script 3 nicht mehr empfohlen und fliegt demnächst komplett aus der Unterstützung. Stattdessen wird empfohlen, dass Pakete `tocbasic` unterstützen. Der Aufwand dafür ist sehr gering und wird mit vielen neuen Möglichkeiten belohnt.

Dieser Hack rüstet die `tocbasic`-Unterstützung für `listings` nach.

`listings`

```

359 <*package & option>
360 \KOMA@ifkey{listings}{\@scrhack@listings}
361 \KOMAExecuteOptions{listings=true}
362 </package & option>

```

```

363 <*package & body>
364 \AfterPackage*{listings}{%
365   \KOMA@key[.scrhack.sty]{listings}{%
366     \PackageWarning{scrhack}{option 'listings' ignored}%
367     \FamilyKeyStateProcessed
368   }%
369   \if@scrhack@listings\scr@hack@load\@pkgextension{listings}\fi
370 }
371 </package & body>

```

`\scr@do@hack@listings` Über dieses Macro wird das Verzeichnis der Listings gesetzt. Die gesamte
`\lstlistoflistings` Funktionalität dafür kann tocbasic überlassen werden.

`\float@addtolists` Diese Anweisung wird nicht länger benötigt und daher auf die ursprüngliche
Definition zurückgesetzt. Da listings ihre Definition mit `\AtBeginDocument`
verzögert, muss dies hier ebenfalls geschehen.

```

372 <*listings & body>
373 \newcommand*{\scr@do@hack@listings}{%
374   \RequirePackage{tocbasic}%
375   \addtotoclist[float]{lol}%
376   \setuptoc{lol}{chapteratlist}%
377   \PackageInfo{scrhack}{redefining \string\lstlistoflistings}%
378   \renewcommand*{\lstlistoflistings}{\listoftoc[\string\lstlistoflistings]{lol}}%
379   \AtBeginDocument{%
380     \scr@ifexpected{\float@addtolists}{%
381       \def\float@addtolists##1{\addtocontents{lol}{##1}}%
382     }{%
383       \PackageInfo{scrhack}{undefining \string\float@addtolists}%
384       \let\float@addtolists\relax
385     }{%
386       \scr@ifexpected{\float@addtolists}{%
387         \def\float@addtolists##1{\addtocontents{lol}{##1}}%
388         \orig@float@addtolists{##1}}%
389       {%
390         \PackageInfo{scrhack}{setting \string\float@addtolists\MessageBreak
391           to \string\orig@float@addtolists}%
392         \let\float@addtolists\orig@float@addtolists
393       }{%
394         \PackageWarningNoLine{scrhack}{unkown \string\float@addtolists\space
395           definition found!\MessageBreak
396           Maybe you are using a unsupported listings version}%
397       }%
398     }%
399   }%
400   \let\scr@do@hack@listings\relax
401 }
402 \scr@ifexpected{\lstlistoflistings}{%

```



```

403 \def\lstlistoflistings{\bgroup
404   \let\contentsname\lstlistlistingname
405   \let\lst@temp\@starttoc \def\@starttoc##1{\lst@temp{lol}}}%
406   \tableofcontents \egroup}%
407 }{%
408 \scr@do@hack@listings
409 }{%
410 \scr@ifexpected{\lstlistoflistings}{%
411   \def\lstlistoflistings{%
412     \begingroup
413     \ifundefined{@restonecoltrue}{}%
414     \if@twocolumn
415       \@restonecoltrue\onecolumn
416     \else
417       \@restonecolfalse
418     \fi
419   }%
420   \float@listhead{\lstlistlistingname}%
421   \parskip\z@\parindent\z@\parfillskip \z@ \@plus 1fil%
422   \@starttoc{lol}%
423   \ifundefined{@restonecoltrue}{}%
424   \if@restonecol\twocolumn\fi
425   }%
426   \endgroup
427 }%
428 }{%
429 \scr@do@hack@listings
430 }{%
431 \PackageWarningNoLine{scrhack}{unknown \string\lstlistoflistings\space
432   definition found!\MessageBreak
433   Maybe you are using a unsupported listings version}%
434 }%
435 }
436 </listings & body>

```

7.7 Der setspace-Hack

Das `setspace`-Paket verwendet `\@ptsize` auf ungünstige Art, indem es davon ausgeht, dass es immer eine ganze Zahl enthält. Das ist aber bei KOMA-Script keineswegs zwingend. Außerdem ist der Wert für `11pt` falsch, weil L^AT_EX in diesem Fall tatsächlich eine 10,95pt-Schrift mit einem Zeilenabstand von 13,6pt einstellt. Damit wäre der korrekte Wert für `\onehalfspacing` beispielsweise:

$$10,95 \text{ pt} \cdot 5/13,6 \text{ pt} \equiv 1,208$$

Tatsächlich stellt `setspace` aber einen Wert von 1,213 ein, was einer effektiven Schriftgröße von 11 pt entsprechen würde. Ebenso stellt es den aktuellen

Abstand bei `\onehalfspacing` nicht relativ zur aktuellen Schriftgröße ein, sondern zur Grundschriftgröße. Damit erhält man bei

```
\documentclass[10pt]{article}
\usepackage{setspace}
\begin{document}
\large\onehalfspacing\raggedright
Fontsize: \csname f@size\endcsname pt\\
Normal baselineskip: \csname f@baselineskip\endcsname\\
baselineskip: \the\baselineskip
\end{document}
```

einen anderen Abstand als bei

```
\documentclass[11pt]{article}
\usepackage{setspace}
\begin{document}
\large\onehalfspacing\raggedright
Fontsize: \csname f@size\endcsname pt\\
Normal baselineskip: \csname f@baselineskip\endcsname\\
baselineskip: \the\baselineskip
\end{document}
```

obwohl beide Male dieselbe Schriftgröße verwendet wird. Streng genommen müsste also bei jeder Änderung der Schriftgröße der Wert Abstand angepasst werden. So weit geht dieser Hack nicht. Stattdessen wird der Wert abhängig von der tatsächlichen Schriftgröße und dem tatsächlichen Basisabstand beim Aufruf der Anweisungen eingestellt. Das ergibt immerhin in den obigen Beispielen gleiche Ergebnisse.

setspace

```
437 <*package & option>
438 \KOMA@ifkey{setspace}{@scrhack@setspace}
439 \KOMAEecuteOptions{setspace=true}
440 </package & option>
441 <*package & body>
442 \AfterPackage*{setspace}{%
443   \KOMA@key[.scrhack.sty]{setspace}{%
444     \PackageWarning{scrhack}{option 'setspace' ignored}%
445     \FamilyKeyStateProcessed
446   }%
447   \if@scrhack@setspace\scr@hack@load\@pkgextension{setspace}\fi
448 }
449 </package & body>
```

`\onehalfspacing` Über diese Anweisung wird der eineinhalbzeilige Satz eingestellt. Ein auf drei Nachkommastellen genauer Wert erscheint mir ausreichend genau.

```

450 <*setspace & body>
451 \scr@ifexpected{\onehalfspacing}{%
452   \long\def\onehalfspacing{%
453     \setstretch{1.25}% default
454     \ifcase \@ptsize \relax % 10pt
455       \setstretch {1.25}%
456     \or % 11pt
457       \setstretch {1.213}%
458     \or % 12pt
459       \setstretch {1.241}%
460     \fi
461   }%
462 }{%
463   \renewcommand*\onehalfspacing{%
464     \@tempdima=\dimexpr (\f@size pt)*1500/
465                       (\dimexpr \f@baselineskip\relax)*\p@/1000\relax
466     \expandafter\setstretch\expandafter{\strip@pt\@tempdima}%
467   }%
468 }{%
469   \PackageWarning{scrhack}{unknown \string\onehalfspacing\space
470     definition found!\MessageBreak
471     Maybe you are using a unsupported setpace version}%
472 }

```

`\doublespacing` Über diese Anweisung wird der zweizeilige Satz eingestellt. Ein auf drei Nachkommastellen genauer Wert erscheint mir ausreichend genau.

```

473 \scr@ifexpected{\doublespacing}{%
474   \long\def\doublespacing{%
475     \setstretch{1.667}% default
476     \ifcase \@ptsize \relax % 10pt
477       \setstretch {1.667}%
478     \or % 11pt
479       \setstretch {1.618}%
480     \or % 12pt
481       \setstretch {1.655}%
482     \fi
483   }%
484 }{%
485   \renewcommand*\doublespacing{%
486     \@tempdima=\dimexpr (\f@size pt)*2000/
487                       (\dimexpr \f@baselineskip\relax)*\p@/1000\relax
488     \expandafter\setstretch\expandafter{\strip@pt\@tempdima}%
489   }%
490 }{%
491   \PackageWarning{scrhack}{unknown \string\doublespacing\space

```

```

492     definition found!\MessageBreak
493     Maybe you are using a unsupported setpace version}%
494 }
495 </setspace & body>

```

7.8 Der lscape-Hack

Das `lscape`-Paket setzt innerhalb der `landscape`-Umgebung die Länge `\textheight` auf den Wert von `\textwidth` obwohl es auf der anderen Seite `\textwidth` nicht auf den Wert von `\textheight` setzt. Das ist inkonsequent. Da David Carlisle außerdem angibt, dass `\textwidth` nicht verändert wird, weil das zu Problemen führen konnte, ist es unverständlich, dass `\textheight` verändert wird, obwohl das ebenfalls zu Problemen führen kann, beispielsweise für `showframe` oder `scrpage`. Daher verändere ich die Definition so, dass auch `\textheight` unverändert bleibt. Dabei muss allerdings auch `pdfscape` berücksichtigt werden. Das ist am einfachsten mit `xpatch`.

`lscape`

```

496 <*package & option>
497 \RequirePackage{xpatch}%
498 \KOMA@ifkey{lscape}{@scrhack@lscape}%
499 \KOMAEecuteOptions{lscape=true}%
500 </package & option>
501 <*package & body>
502 \AfterPackage*{lscape}{%
503   \if@scrhack@lscape\scr@hack@load\@pkgextension{lscape}\else
504     \KOMA@key[.scrhack.sty]{lscape}{%
505       \PackageWarning{scrhack}{option 'lscape' ignored}%
506       \FamilyKeyStateProcessed
507     }%
508   \fi
509 }
510 </package & body>

```

`\landscape` Über diese Anweisung wird die `landscape`-Umgebung von `lscape` gestartet. Genau diese muss gepatcht werden. Dafür wird das Paket `xpatch` benötigt. Da der Patch nur geladen wird, wenn die Option dafür gesetzt ist, kann die Option daher nur ein- und ausgeschaltet werden, wenn sie bis zum Laden des Pakets aktiviert wurde.

```

511 <*lscape & body>
512 \xpatchcmd{\landscape}{\textheight=\vsize}{%
513   \if@scrhack@lscape

```

Es gibt allerdings in der Tat eine Stelle, an der ein verändertes Wert von `\textheight` benötigt wird. Das ist wenn innerhalb von `\@outputpage` der Wert von `\@colht` reinitialisiert wird. Also wird das entsprechend auch noch hinein gepatcht.

```

514   \scrh@LT@textheight=\vsize
515   \let\scrh@LT@outputpage\@outputpage
516   \def\@outputpage{\scrh@LT@outputpage\global\@colht\scrh@LT@textheight}%
517   \else
518     \textheight=\vsize
519   \fi
520 }{%
521   \PackageInfo{scrhack}{\string\landscape\space patched to make
522     \string\textheight\space change optional}%
523 }{%
524   \PackageWarning{scrhack}{Cannot patch \string\landscape!\MessageBreak
525     Maybe you are using a unsupported lscape version}%
526   \@scrhack@lscapefalse
527 }

```

`\scrh@LT@textheight`

```

528 \newlength{\scrh@LT@textheight}
529 \lscap & body

```

7.9 Optionen ausführen

Zum Schluss noch die Optionen ausführen. Im Paket wird diese Anweisung allerdings vor den Anweisungen der Hacks und den Anweisungen aus dem Abschnitt »Verwendete Anweisungen« stehen.

```

530 <*package & option>
531 \KOMAProcessOptions\relax
532 </package & option>

```

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		D	
<code>\@schapter</code> <u>64</u>	<code>\DeclareNewFloatType</code> <u>283</u>
<code>\@spart</code> <u>64</u>	<code>\doublespacing</code> <u>473</u>
<code>\@ssect</code> <u>64</u>		

F		Optionen:	
float (Option)	<u>168</u>	floatrow	<u>270</u>
\float@addtolists ..	<u>181</u> , <u>283</u> , <u>372</u>	float	<u>168</u>
floatrow (Option)	<u>270</u>	hyperref	<u>32</u>
H		listings	<u>359</u>
hyperref (Option)	<u>32</u>	lscape	<u>496</u>
L		setspace	<u>437</u>
\landscape	<u>511</u>	S	
listings (Option)	<u>359</u>	\scr@do@hack@listings	<u>372</u>
\listof	<u>181</u> , <u>283</u>	\scr@hack@load	<u>12</u>
lscape (Option)	<u>496</u>	\scr@ifexpected	<u>1</u>
\lstlistoflistings	<u>372</u>	\scrh@LT@textheight(Länge) ..	<u>528</u>
N		secnumdepth (Zähler)	<u>1</u>
\newfloat	<u>181</u>	setspace (Option)	<u>437</u>
O		Z	
\onehalfspacing	<u>450</u>	Zähler:	
		secnumdepth	<u>1</u>