Informations supplémentaires sur les principales classes scrbook, scrreprt et scrartcl et sur le paquet scrextend

Ce chapitre fournit des précisions supplémentaires sur les classes de KOMA-Script Scrbook, Scrreprt et Scrartcl, mais aussi des instructions concernant des fonctions disponibles pour le pack scrextend. Certaines parties de ce chapitre ont trait à KOMA-Script book[KM12]. Cela n'est pas un problème, pour un utilisateur qui applique juste les classes et n'a pas besoin de ces informations. Certaines sont destinées à l' usager qui résout des tâches inhabituelles ou veut écrire ses propres classes en utilisant KOMA-Script. Une autre partie de l'information examine la compatibilité de ces classes avec les classes LaTeX standard ou des versions existantes antérieures de KOMA-script. Des pièces existent seulement pour la compatibilité avec les versions antérieures de KOMA-Script. Elles sont écrites dans une police sans serif et ne doivent plus être utilisées.

Vous trouverez à ce sujet plus d'informations dans le livre KOMA-Script[KM12].

21.1. Compléments d'information à propos du mode d'emploi

Vous trouverez à ce sujet des d'informations dans le livre KOMA-Script[KM12].

21.2. Interaction de KOMA-Script avec d'autres packs

Vous trouverez à ce sujet des d'informations dans le livre KOMA-Script[KM12].

21.3. Instructions pour experts

Dans cette section, les instructions décrites ont, pour l'utilisateur moyen, peu ou pas d'intérêt, mais ces commandes offrent des fonctionnalités supplémentaires pour les experts. Puisque l'information s'adresse aux experts elle est condensée.

\KOMAClassName
\ClassName

\KOMAClassName stocke le nom de la classe KOMA- Script actuellement utilisée. Quelqu'un qui veut savoir si une classe KOMA-Script est utilisée ou quel KOMA-Script est employé, peut facilement le tester avec cette commande. En revanche, \ClassName indique la classe standard qui est remplacée par la classe KOMA-Script.
Noter que l'existence de \KOMA-Script n'indique pas nécessairement l'utilisation d'une classe KOMA - Script. Non seulement les classes KOMA-Script mais aussi tous packs KOMA-Script définissent \KOMA-Script. En outre d'autres packs peuvent également juger opportun de définir le pictogramme KOMA-Script avec ce nom.

\addtocentrydefault{niveau}{numéro}{titre}

Pour créer une table des matières, KOMA-Script n'utilisent pas \addcontentsline directement, mais appelle \addtocentrydefault avec des arguments similaires. La commande peut être utilisée à la fois, pour les deux entrées numérotée et sans numéro. Ainsi chaque niveau est celui d'organisation du texte, c'est-à-dire partie, chapitre, section, subsection, subsubsection, paragraphe ou alinéa. Le nombre de

sections déjà formatées est donné par le deuxième argument, le numéro qui peut être vide. Le texte de l'entrée est donnée par l'argument rubrique. Il est prescrit de protéger les commandes fragiles intérieures dans cet argument avec \protect.

Pour l'argument numéro existe une fonction spéciale. Un argument vide signale que l'entrée sans numéro doit être générée. Par défaut, ce sera réalisé avec

```
\addcontentsline{toc}{niveau}{%
   \protect\nonumberline titre%
}
```

Toutefois, si l'argument n'est pas vide une entrée numérotée sera générée dont le numéro sera celui de l'article préformaté. Par défaut, utiliser koma - Script

```
\addcontentsline{toc}{niveau}{%
 \protect \numberline{numéro}titre%
}
```

pour la génération de cette entrée.

Les auteurs de pack et les auteurs de classes wrapper peuvent redéfinir cette déclaration pour manipuler les entrées. Ainsi, on pourrait suggérer par exemple :

```
\renewcommand{\addtocentrydefault}[3]{%
\ifstr{#3}{}{%
}{%
\ifstr{#2}{}{%
\addcontentsline{toc}{#1}{\protect\nonumberline#3}%
}{%
\addcontentsline{toc}{#1}{\protect\numberline{#2}#3}%
}%
}%
}%
```

indispensable pour s'assurer que les articles n'ont pas d'entrée avec un titre vide. Dans la pratique, un tel changement n'est pas nécessaire car la suppression des entrées vides est déjà traitée d'autres manières dans les classes de KOMA-Script. Voir la description des structures de commandes dans la section 3.16.

\addparttocentry{*Numéro*}{titre}
\addchaptertocentry{*Numéro*}{titre}
\addsectiontocentry{*Numéro*}{titre}
\addsubsectiontocentry{*Numéro*}{titre}
\addsubsubsectiontocentry{*Numéro*}{titre}
\addparagraphtocentry{*Numéro*}{titre}
\addsubparagraphtocentry{*Numéro*}{titre}

La déclaration \addtocentrydefault documentée ci-dessus est appelée directement par les classes de KOMA-Script si aucune commande individuelle pour le niveau spécifié n'a été définie ou si cette commande est \relax.
Par défaut, les instructions données sont toutes définies de façon à transmettre leur niveau et les arguments directement à \addtocentrydefault.

\raggedchapterentry

Avec l'aide de \raggedchapterentry, il y a la possibilité de créer des entrées de chapitre à la place de la table des matières (TOC) dans une composition avec flottants préréglés justifiés à gauche. Cette déclaration doit être définie comme \raggedright. Vous pouvez aussi consulter les instructions de limitation dans le paragraphe 21.2 – avertissement. Cependant, la définition prédéfinie est vide par défaut. D'autres paramètres \raggedchapterentry ne sont pas recommandés et peuvent provoquer des résultats inattendus

\@fontsizefilebase

Le préfixe scrsize spécifié pour les tailles de police à la section 21.1 est simplement le réglage par défaut de la macro interne \fontsizefilebase utilisée lorsque la macro n'a pas été définie lors du chargement d'une classe KOMA-Script ou du pack scrextend. Les auteurs de classes wrapper peuvent définir cette macro avec un autre préfixe pour utiliser complètement différents fichiers de taille de police. Ils peuvent aussi modifier ou désactiver la solution de repli en redéfinissant \changefontsizes. Cette macro ne supporte qu'un seul argument : la taille désirée de la police voulue.

\newkomafont[Avertissement]{article}{par default}
\alias coma police{alias}{article}

Les experts peuvent utiliser \newkomafont afin de définir une valeur par défaut pour le style de police d'un élément. Par la suite, ce « par défaut » peut être modifié à l'aide des commandes \setkomafont et \addtokomafont (voir section 3.6.).
Bien sûr, ce n'est pas assez pour utiliser le style de police défini. L'expert lui-même doit préparer son code pour utiliser la commande \usekomafont et l'appliquer dans la définition des codes de cet élément.

L'argument optionnel définit un message d'avertissement qui sera affiché chaque fois que le style de police par défaut de cet élément sera changé. L'expéditeur de l'avertissement dans ce cas sera la classe de KOMA-Script utilisée ou le pack scrextend. La commande \aliaskomafont définit un alias pour un élément existant déjà défini.

KOMA-Script informera l'utilisateur automatiquement sur le vrai nom de l'élément,. Un nom d'alias peut être utilisé, par exemple, si un développeur trouve un meilleur nom pour un élément, qui a été défini précédemment avec un autre nom, et si cet ancien nom doit rester utilisable en raison de la compatibilité. Aussi un alias peut augmenter une facilité d'utilisation. KOMA-Script lui-même fait une riche utilisation de cette opportunité.

\addkomafontrelaxlist{Makro}

Comme expliqué dans la partie I du manuel, seules les commandes pour sélectionner la taille, la famille, le codage, l'épaisseur de la ligne, la forme et la couleur peut être contenues dans les paramètres des éléments de la police. Ainsi, un changement de couleur en latex n'est pas très transparent et peut entraîner des effets indésirables lorsque \usekomafont est utilisé à un emplacement défavorable.

Les utilisateurs ont tendance à écrire dans les packs des choses tout à fait différentes, parfois très critiquables dans les paramètres de polices, par exemple un \MakeUppercase à la fin d'un réglage. En usage interne, il est possible que beaucoup de ces paramètres, théoriquement interdits, fonctionnent normalement, même si la dernière commande de la police, mise en argument, doit par exemple utiliser \textbf à la place de \bfseries. Aucune garantie n'existe cependant.

Dans certains cas, il est nécessaire au sein de KOMA-Script, vraiment limiter le passage à des paramètres de police. Ceci a lieu par exemple avec \usefontofkomafont au lieu de \usekomafont (voir la section 3.6).

Les commandes \usefontofkomafont et apparentées ont leurs limites. Par conséquent, les paramètres de réglage d'un élément de police ne doivent pas s'attendre à un argument présumé entièrement extensible. Ce qui est exactement le cas, par exemple, dans \MakeUppercase. Par conséquent KOMA-Script maintient une liste interne de macros qui se trouvent dans \relax de \usefontofkomafont et apparentés. Par défaut, ceux-ci comprennent, entre autres, \color \normalcolor, \MakeUppercase et \MakeLowercase. D'autres instructions peuvent être ajoutées individuellement avec \addtokomafontrelaxlist.

Il convient de noter que la macro spécifiée est définie exclusivement pour \relax. Tous les arguments s'appliquant à la police sont donc éventuellement exécutés localement. Par conséquent et en aucune façon, les déclarations comme \setlength ne peuvent être ajoutées à cette liste. L'utilisateur est seul responsable de toutes les erreurs qui pourraient être causées par l'usage de \addtokomafontrelaxlist. En outre, cette possibilité ne doit pas être interprétée comme une légitimation pour ajouter toutes sortes d'instructions aux paramètres de la police !

\IfExistskomafont{Element}{Code}{Sinon}

Depuis les écritures, certains éléments peuvent être modifiés uniquement à partir de certaines versions de KOMA-Script, il est parfois utile de pouvoir pré-tester un élément pour voir si cette possibilité existe. L'instruction exécute alors le code si et seulement si l'article via \newkomafont ou \aliaskomafont était définis et donc setkomafont avec \setkomafont ou \addtokomafont changé et le \use ... komafont instructions peuvent être interrogés. Dans le cas contraire, le code sinon est exécuté.

\setparsizes{tiret}{distance}{ligne de fin d'espace vide}

Cette déclaration permet à la fois le retrait et l'espacement de paragraphe, mais aussi d'ajuster l'espace libre à la fin de la dernière ligne de chaque paragraphe de scrbook, scrreprt ou scrartcl. Cette commande doit être utilisée chaque fois que des changements devraient aussi être reconnus par l'option parskip=relative. KOMA-Script lui-même utilise cette commande, par exemple, avec

\setparsizes{0pt}{0pt}{0pt plus 1fil}

pour désactiver à la fois le retrait de paragraphe, la distance entre paragraphes et laisser un espace blanc à la fin de la dernière ligne de paragraphe. Une telle

mesure est utile si un paragraphe ne se compose que d'une boîte, qui est réglée sans un écart vers le haut ou vers le bas et occupe toute la largeur de la colonne. Si la boîte ne doit pas s'étendre sur toute la largeur et doit être imprimée avec les paramètres actuels de retrait et d'espace entre les paragraphes, l'utilisation de

\setlength{\parfillskip}{0pt plus 1fil}

serait préférable et recommandée.

Un nouveau calcul ou la réactivation des paramètres de l'empagement et les marges (voir chapitre 2) se traduit toujours depuis KOMA-Script 3.17 par un réajustement des valeurs via \setparsizes si ces valeurs n'ont pas changé dans l'intervalle.

Cela devrait être une raison de plus de ne pas changer les réglages passés pour KOMA-Script. Le recalcul sera désactivé avec un paramètre de compatibilité à une version antérieure (voir la section 3.2, selon la version de l'option).

Pour plus d'informations, consulter le livre KOMA-Script [Koh14a].

\DeclareSectionCommand[paramètres]{nom}
\DeclareNewSectionCommand[paramètres]{nom}
\RedeclareSectionCommand[paramètres]{nom}
\ProvideSectionCommand[paramètres]{nom}

Avec ces instructions, une nouvelle structure de commande \Name peut être définie ou une structure de commande \Name existante être modifiée. Ces réglages sont effectués avec arguments optionnels dont les paramètres séparés par des virgules, sont dans une liste de clés de type key = valeur allouée.

En plus indépendant du style des fonctionnalités phares qui peuvent être trouvées dans le Tableau 21.1, il existe également des propriétés qui dépendent du style particulier. Actuellement, les styles suivants sont disponibles :

chapter :
Style d'un titre de chapitre.

De nouvelles instructions dans ce style ne peuvent pas, jusqu'ici, être définies, mais il est possible de reconfigurer uniquement la commande
\chapter existante. Les propriétés du Tableau 21.3 sont disponibles. La commande \addchap ainsi que les formes étoilées sont automatiquement reconfigurées avec \chapter et ne peuvent être modifiées indépendamment. Il est à noter que ce type n'est pas fourni avec scrartcl.

Part :
Style de Part.

De nouvelles instructions dans ce style ne peuvent pas, à ce jour, être définies, mais il est possible de reconfigurer la commande \part existante. Les propriétés du Tableau 21.3 sont disponibles. La commande \addpart ainsi que les formes étoilées sont automatiquement reconfigurées avec \part et ne peuvent être modifiées indépendamment.

Section :
Style d'un titre de section.

Ce style est utilisé actuellement pour \section, \subsection, \subsubsection, \paragraph et \subparagraph. De nouvelles rubriques peuvent être définies dans ce style. Les propriétés du Tableau 21.2 sont disponibles pour configurer des rubriques existantes ou nouvelles. Redéfinir le style des clés, afterskip, beforeskip, font, indent, level, tocindent et tocnumwidth est obligatoire
Ceci vaut également si une commande n'est pas, jusqu'ici une commande du dispositif redéfinie avec \RedeclareSectionCommand. La commande \addsec, ainsi que les formes étoilées sont reconfigurées automatiquement avec \section et ne peuvent être modifiées indépendamment. Lorsque vous définissez une structure de commandement dans ce style, un élément est créé, dans le même temps, portant le même nom, avec les paramètres de police \setkomafont et \addtokomafont (voir la section 3.6) qui peuvent être modifiés, si un tel élément n'existe pas.

\DeclareNewSectionCommand définit une nouvelle commande. Si le même nom est déjà utilisé par TeX, l'erreur est signalée et il n'y a pas redéfinition.

\ProvideSectionCommand se comporte de manière similaire, mais ne donne aucun message d'erreur. \RedeclareSectionCommande, cependant, ne peut être utilisé pour modifier une assignation existante à un commande du dispositif avec les propriétés spécifiées. Il ne vérifie pas si \Name était auparavant une commande de division mais il ne doit y avoir qu'un seul nom déjà occupé par TEX.

Avec \DeclareSectionCommand aucune vérification n'est faite pour savoir si Name est déjà utilisé ailleurs. Au lieu de cela, la commande \Name définit strictement les propriétés spécifiées.

Chaque division comprend également une contre-déclaration du même nom, qui est créée telle que requis par les quatre nouvelles commandes avec \newcounter. Il en va de même pour la sortie formater du compteur, \theName, le formatage du compteur, \Name format, l'instruction pour créer un titre de colonne \Name mark, utilisé pour le le formatage du compteur \Name markformat, le nom de l'objet, et le niveau de compteur, \Name numdepth.

Les instructions pour la création d'un titre de colonne \Name mark est, le cas échéant, pré-défini si aucun titre de colonne n'est créé. La sortie du compteur, \theName est prédéfinie avec chiffre arabe. Si la clé à l'intérieur du compteur est définie comme dépendante d'un autre compteur, alors l'édition de cet autre compteur est précédée par un point.

Tableau 21.1:
Clés et valeurs possibles des propriétés dans la déclaration de commandes du dispositif

Clé	Valeur	Signification
Counterwithin	**Nom du compteur**	Le niveau hiérarchique d'appartenance du compteur est dépendant de sa valeur spécifiée. Si son nom change de niveau sur \stepcounter ou \refstepsounter, le niveau de détail pour le compteur associé est remis à 0. En outre, \theZählername qui suit est précédé par un point dans la question du niveau hiérarchique de son appartenance.
expandtopt	**Interrupteur**	Si cet interrupteur est actif, toutes les valeurs de réglage des longueurs indiquées ci-dessous sont pleinement développées, évaluées et stockées converti en pt. Si le commutateur n'est pas actif, alors les valeurs sont uniquement utilisées à des fins d'essais et d'évaluation. Elles comprennent les valeurs de simples interrupteurs du tableau 2.5.
level	**Entier**	Valeur numérique du niveau de détail (voir compteur secnumdepth; 3.16), la valeur doit être unique.
style	**Nom**	Indique le style de rubriques.
tocindent	**Longueur**	L'alimentation horizontale des commandes sectionnelles appartenant à l'entrée dans la table des matières, produites en réponse à tocdepth (voir section 3.9)
toclevel	**Entier**	Le niveau de la structure de commandement appartenant à l'entrée dans la table des matières, si elle est de niveau différent (voir également tocdepth à la section 3.9).
tocnumwidth	**Longueur**	La largeur qui est réservée pour le numéro de section dans l'entrée de commande-cadre appartenant à la table des matières.

Tabelle 21.2 :
Clés supplémentaires et valeurs des propriétés dans la déclaration des commandes du style de la section

afterskip	**Longueur**	Une valeur négative signifie que le texte d'en-tête doit être formaté comme marque à partir du haut. La quantité dans ce cas indique la distance horizontale à la rubrique. Une valeur positive, cependant, se traduit par une distance verticale après le titre.
beforeskip	**Longueur**	La quantité indique la distance verticale en face de la position. Si la valeur est négative, un espace positif sera néanmoins inséré. Les valeurs négatives signifient dans ce cas que le paragraphe devrait être placé après le titre sans alimentation horizontale.

font	**Réglage de la police**	Paramètres de police qui doivent être utilisés en plus de l'élément disposition sur la question du titre. Tous ceux permettant l'usage de \setkomafont et \addtokomafont sont admis.
indent	**Longueur**	Indentation de la marge de gauche avant le nombre et le texte du titre

Tableau 21.3 : Clés possibles et valeurs des propriétés dans la répartition de la configuration des commandes du style de chapitre

Clé	Valeur	Signification
afterskip	**Longueur**	La quantité est la distance verticale à la rubrique.
beforeskip	**Longueur**	La quantité indique la distance verticale en face de la position. Si la valeur est négative, un espace positif sera néanmoins inséré. Les valeurs négatives signifient dans ce cas que le paragraphe devrait être placé après le titre sans entrée horizontale
font	**Réglage de la police**	Paramètres de police qui doivent être utilisés en plus de la disposition de l'élément du titre. Tous ceux permettant l'usage de \setkomafont et \addtokomafont sont admis.
innerskip	**Longueur**	Distance verticale entre la ligne de préfixe et la position du texte, si une ligne de préfixe est utilisée.
prefixfont	**Réglage de la police**	Paramètres de police, en plus de l'élément de programmation et l'élément de structure de commande pour être utilisés dans la production d'une ligne de préfixe dans l'en-tête. Tous ceux permettant l'usage de \setkomafont et \addtokomafont sont admis.

Exemple :
Pour une raison quelconque, la position de \paragraph ne devrait plus être redéfinie par la plus haute marque (spitzmark), mais comme une rubrique telle que \subsubsection, à une distance de 10pt au-dessus et aucun espacement supplémentaire ne sera inséré au-dessous de la rubrique. Ce pourrait être possible avec :

```
\RedeclareSectionCommand[%
beforeskip=-10pt,%
afterskip=1sp%
]{paragraph}
```

La valeur négative avant la distance verticale au-dessus du titre est générée avec beforeskip, et dans le même temps, l'alimentation de la première section est arrêtée après l'intitulé. Après le titre, aucun espace vertical n'est nécessaire, signalé ici par la valeur 1 sp. La raison en est simple : une valeur de 0 pt n'est pas considérée, à ce

point, comme une valeur positive par LaTeX, et génère ainsi un en-tête sous une forme de marque haute. La plus petite valeur positive est de 1 sp.
En règle générale, pour la compensation verticale (voir \flushbottom, Section 3.4) il vaut mieux prévoir des distances avec une certaine marge de manœuvre et éviter de se faire surprendre soi-disant par erreur :

```
\RedeclareSectionCommand[%
beforeskip=-10pt plus -2pt minus -1pt,%
afterskip=1sp plus -1sp minus 1sp%
]{paragraph}
```

Il convient de noter que l'on peut se faire piéger lorsque le signe de changement de distance verticale, donc avec beforeskip dans l'exemple, est négatif. Dans le même temps la possibilité vous est donnée de vous assurer que la distance, avec afterskip, peut se réduire effectivement à 0.

Dans l'exemple, en fait seules les clés beforeskip et afterskip devraient être utilisées, parce que depuis KOMA-Script v3.15, \paragraph est déjà défini en interne par \DeclareSectionCommand et donc le reste des paramètres peut être utilisé inchangé. La définition originale de \paragraph correspond avec scrartcl :

```
\DeclareSectionCommand[% level=4,
indent=0pt,
beforeskip=3.25ex plus 1ex minus .2ex, afterskip=-1em,
font={},
tocindent=7em,
tocnumwidth=4.1em,
counterwithin=subsubsection
]{paragraph}
```

Dans scrreprt et scrbook, des valeurs partiellement dissidentes sont utilisées.

Tableau 21.4 : Clés possibles et valeurs des propriétés dans la répartition de la configuration des commandes du style de part

Clé	Valeur	Signification
afterskip	**Longueur**	La quantité est la distance verticale à la rubrique.
beforeskip	**Longueur**	La quantité indique la distance verticale en face de la position. Si la valeur est négative, un espace positif sera néanmoins inséré. Les valeurs négatives signifient dans ce cas que le paragraphe devrait être placé après le titre sans entrée horizontale
font	**Réglage de la police**	Paramètres utilisés pour la police du titre, en plus de la disposition. Tous ceux permettant l'usage de \setkomafont et \addtokomafont sont admis.
innerskip	**Longueur**	Distance verticale entre la ligne de préfixe et la position du texte, dans scrbook et scrreprt

prefixfont Réglage de la police Paramètres de police, en plus de l'élément de programmation et de structure de commande utilisés dans la production d'une ligne de préfixe dans l'en-tête. Tous ceux permettant l'usage de \setkomafont et \addtokomafont sont admis.

Pour \chapter quelques options pour les titres des articles (voir Section 3.16) dépendent des rubriques. Ces paramètres spécifiques peuvent être trouvés dans le Tableau 21.5. Le Tableau 21.6 fournit un aperçu de tous les préréglages. Il convient de noter que la taille 1ex, prédéfinie du titre ou d'entrée de la table des matières dépendent de \baselineskip.

Tableau 21.5 : Instructions prédéfinies pour l'espacement vertical dans les têtes de chapitre de scrbook et scrreprt selon l'option rubriques

Avec headings=big:

Réglage	**valeur par défaut**
afterskip	1.725\baselineskip plus .115\baselineskip minus .192\baselineskip
beforeskip	3.3\baselineskip+\parskip
font	\huge

Avec headings=normal:

afterskip	1.5\baselineskip plus .1\baselineskip minus .167\baselineskip
beforeskip	3\baselineskip+\parskip
font	\LARGE

Avec headings=small:

afterskip	1.35\baselineskip plus .09\baselineskip minus .15\baselineskip
beforeskip	2.8\baselineskip+\parskip
font	\Large

Tabelle 21.6 : Préférences pour les rubriques de scrbook et scrreprt

\part :

Réglage	**valeur par défaut**
afterskip	0pt plus 1fil
beforeskip	0pt plus 1fil + \baselineskip
font	voir l'élément part, Tableau 3.15
innerskip	20pt
level	-1
prefixfont	voir l'élément partnumber, Tableau 3.15
tocindent	0pt
tocnumwidth	2em

\chapter :

afterskip	siehe tabelle 21.5
beforeskip	siehe tabelle 21.5
font	voir l'élément chapter, Tableau 3.15
innerskip	0pt
level	0
prefixfont	voir l'élément chapterprefix, Tableau 3.15
tocindent	0pt
tocnumwidth	1.5em

\section :

Réglage	**valeur par défaut**
afterskip	2.3ex plus .2ex
beforeskip	-3.5 plus -1ex minus -.2ex
font	voir l'élément section, Tableau 3.15
indent	0pt
level	1
tocindent	1.5em
tocnumwidth	2.3em

\subsection :

afterskip	1.5ex plus .2ex
beforeskip	-3.25 plus -1ex minus -.2ex
font	voir l'élément subsection, Tableau 3.15
indent	0pt
level	2
tocindent	3.8em
tocnumwidth	3.2em

\subsubsection :

afterskip	1.5ex plus .2ex
beforeskip	-3.25 plus -1ex minus -.2ex
font	voir l'élément subsubsection, Tableau 3.15
indent	0pt
level	3
tocindent	7.0em
tocnumwidth	4.1em

\paragraph :

afterskip	-1em
beforeskip	3.25 plus 1ex minus .2ex
font	voir l'élément paragraph, Tableau 3.15
indent	0pt
level	4
tocindent	10em
tocnumwidth	5em

\subparagraph :

afterskip	-1em
beforeskip	3.25 plus 1ex minus .2ex
font	voir l'élément subparagraph, Tableau 3.15
indent	\scr@parindent
level	5
tocindent	12em
tocnumwidth	6em

Ces instructions décrivent un certain nombre de commandes qui peuvent être également définies ou modifiées. La liste est une liste de noms de commandes de sectionnement séparées par des virgules.

Ces instructions diffèrent dans deux autres points des consignes ci-dessus pour définir ou modifier une structure de commande unique.

Tout d'abord, en cas d'échec, soit quand une déclaration existait auparavant dans \DeclareNewSectionCommands ou alors, bien que définie, n'existait pas encore dans \RedeclareSectionCommands. Une erreur correspondante sera, bien sûr, émise en conséquence.

D'autre part, il existe un autre paramètre, increaselevel = entier. Ainsi, l'importance de level et toclevel (voir tableau 21.4) va changer dans la mesure où leurs valeurs ne sont que des paramètres de la première commande de sectionnement de la liste de nom. Pour toutes les commandes de sectionnement autres, les valeurs de level et toclevel sont augmentées de la valeur de increaselevel. Le réglage utilisé est celui de increaselevel sans affectation de valeur, par défaut la valeur 1.

\chapterheadstartvskip
\chapterheadmidvskip
\chapterheadendvskip
\partheadstartvskip
\partheadmidvskip
\partheadendvskip
partheademptypage

Ces énoncés sont utilisés à l'intérieur de la définition des rubriques \chapter, \part, \addchap, \addpart et de leurs variantes étoilées. Ainsi, \chapterheadstartvskip est une déclaration qui vise à ajouter une distance verticale avant l'en-tête de chapitre, tandis que \chapterheadendvskip vise à l'ajouter après le titre du chapitre.
Les commandes \partheadstartvskip et \partheadendvskip sont utilisées pour insérer des espaces verticales au-dessus et au-dessous des titres de rubriques. Dans ce cas, un saut de page est interprété comme faisant partie de la distance verticale et, dans scrbook et scrreprt, \partheadendvskip est inclus par défaut.

Les paramètres par défaut pour ces sept déclarations ne dépendent pas, depuis KOMA-Script 3.15, du réglage des positions d'options (voir section 3.16). Les définitions des têtes de chapitre de KOMA-Script 3.17 correspondance à :

```
\newcommand*{\chapterheadstartvskip}{\vspace{\@tempskipa}}
\newcommand*{\chapterheadmidvskip}{\par\nobreak\vskip\@tempskipa}
\newcommand*{\chapterheadendvskip}{\vskip\@tempskipa}
```

Diese werden auch bei jeder Verwendung von Option headings=big, headings=normal oder headings=small reaktiviert.

Celles-ci sont réactivées à chaque fois en utilisant l'option rubriques (headings) = big, normal ou small.

La commande \chapter définit automatiquement la longueur interne \@tempskipa avant d'appeler \chapterheadstartvskip avec la valeur résultant du réglage de beforeskip de \DeclareSectionCommand.
La même chose se produit avant d'appeler \chapterheadendvskip avec la valeur du réglage afterskip dans \DeclareSectionCommand, et avant le réglage beforeskip dans \chapterheadmidvskip avec la valeur innerskip dans \DeclareSectionCommand. Avec \chapterheadstartvskip, \chapterheadmidvskip ou \chapterheadendvskip sont redéfinis à volonté et les distances configurables par exemple avec \RedeclareSectionCommand, vous devriez donc également utiliser la nouvelle définition \@tempskipa. Les paramètres

par défaut pour les distances de \part ne dépendent pas de l'option titres, y compris les instructions associées de cette option qui ne sont pas redéfinies. Vos définitions initiales correspondent à scrbook et scrreprt:

```
\newcommand*{\partheadstartvskip}{%
\null\vskip-\baselineskip\vskip\@tempskipa
}
\newcommand*{\partheadmidvskip}{%
\par\nobreak
\vskip\@tempskipa
}
\newcommand*{\partheadendvskip}{%
\vskip\@tempskipa\newpage}
```

und bei scrartcl:

```
\newcommand*{\partheadstartvskip}{%
\addvspace{\@tempskipa}%
}
\newcommand*{\partheadmidvskip}{%
\par\nobreak
}
\newcommand*{\partheadendvskip}{%
\vskip\@tempskipa
}
```

Encore une fois \@tempskipa est réglé avant d'utiliser les commandes au sein de \part en fonction des paramètres de \DeclareSectionCommand.

Étant donné que les distances au-dessus, à l'intérieur et sous des rubriques peuvent être réglées avec la commande \RedeclareSectionCommand beaucoup plus facilement, il est déconseillé, à cet effet, de redéfinir les instructions décrites ici. Cette manipulation doit être réservée à des changements plus radicaux, qui ne sont pas accessibles par \RedeclareSectionCommand. Vous trouverez un exemple sur [KDP] pour placer une ligne au-dessus et en dessous de la tête de chapitre en redéfinissant \chapterheadstartvskip et \chapterheadendvskip.

```
\SecDef{version étoilée}{version normale}
\scr@startsection{nom}{niveau }{entrée}{espace avant}{espace après}
                 {instructions de style}[forme abrégée ]{titre}
\scr@startsection{nom}{niveau }{entrée}{espace avant}{espace après}
                 {instructions de style}*{titre}
\At@startsection{code}
\Before@ssect{code}
\Before@sect{code}
```

Comme expliqué section 3.16 dans la description des commandes Outline, KOMA-Script propose, à l'égard des arguments optionnels de commandes de sectionnement, des options avancées qui en augmente la portée. Pour que ce soit possible, il est nécessaire de remplacer certaines instructions du noyau de latex. Cela comprend les instructions \secdef et \startsection. Pour la signification des paramètres de ces commandes, voir les instructions pour noyau de latex [BCJ+05].

Cependant, ces déclarations sont souvent complètement redéfinies en packs et la fonction de KOMA-Script peut en être partiellement altérée, les déclarations de base de LATEX sont mentionnées mais ne sont tout simplement pas redéfinies, ni les commandes supplémentaires \SecDef et \scr@startsection. Celles-ci peuvent être utilisées par les auteurs de packs comme déclarations de base de latex, mais elles offrent automatiquement la fonctionnalité améliorée de KOMA-Script. Mais ces deux déclarations ne doivent pas être redéfinies, car elles sont susceptibles de changer à tout moment, et la fonctionnalité de KOMA-Script pourrait être affectée, de nouveau, par cette redéfinition.

KOMA-Script utilise en interne \SecDef et \scr@startsection au lieu de \secdef et \@startsection par exemple, dans les commandes Outline qui sont définies avec \DeclareSectionCommand. Une redéfinition ultérieure de \secdef et de \startsection n'a donc pas d'impact sur cette commande de sectionnement.

En remplacement pour les redéfinitions des commandes, KOMA-Script offre la possibilité d'insérer du code supplémentaire dans l'exécution de ses propres extensions. Les codes des appels vers les commandes \At@startsection, \Before@sect et \Before@@sect sont collectés séparément pour des instructions individuelles. Il ne vise pas à supprimer à nouveau le code une fois qu'il est inséré.

Le code de \At@startsection débutera en \scr@startsection immédiatement après l'évaluation du paramètre distance et le calcul de la résultante correspondante \@tempskipa mais appliqué avant d'insérer la distance. Ainsi, il est donc toujours possible de modifier cette distance sur \@tempskipa.

Le code de \Before@sect ou respectivement de \Before@ssect est immédiatement exécuté avant d'appeler \@sect ou \@ssect dans \scr@section. A ce moment, le cas échéant, la distance verticale est déjà insérée avant l'en-tête par \addvspace.

Les commandes \At@startsection, \Before@sect, \Before@ssect sont conçues comme des instructions pour les auteurs de pack et approuvées pour cet usage.

\appendixmore

Dans les classes KOMA-Script, il existe dans la commande \appendix une caractéristique spéciale. Car si \appendixmore est définie, la déclaration \appendix est également effectuée. En interne, cette fonction est utilisée par les classes scrbook et scrreprt de KOMA-Script pour la réalisation de l'option de mise en page appendixprefix (voir Section 3.16). Vous devriez prendre note si vous voulez définir ou redéfinir la macro \appendixmore. Si cette option est utilisée, \newcommand {\appendixmore} {. , , } envoie un message d'erreur pour vous empêcher de changer les options en vigueur, sans même le réaliser.

Exemple :
Vous ne voulez pas que, lorsque vous utilisez la classe scrbook ou scrreprt, le corps du chapitre soit fourni avec un préfixe (voir l'option layout de chapterprefix dans la section 3.16). Pour maintenir cette cohérence, vous ne voulez même pas qu'une telle ligne soit utilisée dans l'annexe. Au lieu de cela, le mot "annexe" devrait être dans les annexes devant la lettre de chapitre, dans la langue appropriée. Cela vaut également

pour le titre et donc, ne pas utiliser l'option de mise en page appendixprefix mais définir dans le document :

```
\newcommand*{\appendixmore}{%
  \renewcommand*{\chapterformat}{%
    \appendixname~\thechapter\autodot\enskip}%
  \renewcommand*{\chaptermarkformat}{%
    \appendixname~\thechapter\autodot\enskip}}
```

Si plus tard, vous décidez d'utiliser encore l'option appendixprefix, vous obtiendrez, en raison de la commande déjà définie \appendixmore, un message d'erreur. Il doit empêcher que la définition inaperçue ci-dessus ne remplace les paramètres ajoutés à l'option.

Wenn Sie ein vergleichbares Verhalten des Anhangs für die Klasse scrartcl erreichen wollen, so ist dies ebenfalls möglich. schreiben Sie dazu beispielsweise folgendes in die Präambel Ihres Dokuments :

Il est aussi possible d'obtenir un comportement similaire pour l'annexe de la classe scrartcl, en écrivant dans le préambule de votre document :

```
\newcommand*{\appendixmore}{%
  \renewcommand*{\sectionformat}{%
    \appendixname~\thesection\autodot\enskip}%
  \renewcommand*{\sectionmarkformat}{%
    \appendixname~\thesection\autodot\enskip}}
```

Pour l'explication de cet exemple de redéfinition de commande, voir la section 3.16.

\newbibstyle[*style parent*]{*nom*}{*instructions*}
\newblock
\@openbib@code
\bib@beginhook
\bib@endhook

Les classes standard fournissent déjà la commande \newblock pour la structuration des entrées de bibliographie. Le but exact de cette commande dépend des options de classe. Utiliser l'option openbib redéfinit les commandes \@openbib@code et \newblock à la fin de la classe standard utilisée. Parmi les classes standard, la commande \@openbib@code est exécutée au début de la liste pour la bibliographie, plus précisément dans la détermination des paramètres de cette liste. On peut supposer que les nombreux packs qui redéfinissent la bibliographie exécutent cette instruction en conséquence.

Dans les classes de KOMA-Script quelque chose de similaire se produit. Toutefois, \@openbib@code n'est pas redéfini à la fin de la classe. Au lieu de cela, le style openstyle est défini avec \newbibstyle pour la bibliographie. Les instructions données lors de la mise en œuvre comprennent les redéfinitions appropriées de \@openbib@code et de \newblock . Si maintenant l'option bibliography=openstyle est sélectionnée, les instructions pour redéfinir \@openbib@code et \newblock seront exécutées immédiatement.

En plus de \@openbib@code et de \newblock les instructions peuvent aussi être redéfinies dans \bib@beginhook et \bib@endhook. La commande \bib@beginhook sera exécutée immédiatement après le titre et le préambule de la bibliographie, mais avant la liste des entrées de bibliographie. La commande \bib@endhook est exécutée immédiatement après la liste à la fin de la bibliographie. Ces commandes seront exécutées au début et la fin de chaque partie de la bibliographie, soit juste avant et juste après \BreakBibliography, dans le cas où ce dernier (voir la section 23.3) interfère dans la bibliographie.

Les commandes \newblock, \@openbib@code, \bib@beginhook et \bib@endhook auront une définition vide au début de chaque nouveau style de bibliographie. Après cela, les instructions de l'option spécifiée dans la définition du style parent de la bibliographie, puis les instructions du style de la bibliographie elle-même seront exécutées. Il s'ensuit également que chacune des quatre instructions ne doit en aucune manière être définie avec \newcommand mais, au besoin, redéfinie avec \renewcommand.

L'utilisateur définit avec les commandes \AfterBibliographyPreamble et \AtEndBibliography des instructions supplémentaires pour l'exécution après le préambule et à la fin de la bibliographie. L'instruction \AfterBibliographyPreamble sera exécutée une seule fois au début de la bibliographie, mais après les instructions de \bib@beginhook et l'instruction de \AtEndBibliography exécutée une seule fois à la fin de la bibliographie, mais avant les instructions de \bib@endhook.

Le pack multicol (voir[Mit00]) pourrait être utilisé afin de définir un style de bibliographie pour une impression de cette dernière avec deux colonnes:

```
\newbibstyle{twocolumstyle}{%
\renewcommand*{\bib@beginhook}{\begin{multicols}{2}}%
\renewcommand*{\bib@endhook}{\end{multicols}}}%
```

Pour proposer une variante ouverte de ce style, on peut utiliser la puissance de l'héritage et le spécifier quand vous créez un style avec les parents :

```
\newbibstyle{twocolumopenstyle}[openstyle]{%
\renewcommand*{\bib@beginhook}{\begin{multicols}{2}}%
\renewcommand*{\bib@endhook}{\end{multicols}}}%
```

Ces nouveaux styles définis peuvent être sélectionnés, comme d'habitude, en utilisant l'option bibliographie.

21.4. Options et commandes plus ou moins obsolètes

Pour plus d'informations, consulter le livre KOMA-Script [KM12].