

THE LANGSELECT PACKAGE

VO.2.1 2025/09/04

Generierung verschiedener Sprachversionen aus einer Quelle

Matthias WERNER¹

<https://github.com/tuc-osg/osglecture>

Dieses Paket unterstützt die Generierung von Dokumenten in zwei oder drei Sprachvarianten aus einem gemeinsamen Quelldokument. Das Paket gehört zum `osglecture`-Bundle und wird in die `osglecture`-Klasse integriert und vom `Buildscript o11m` unterstützt, kann aber auch eigenständig genutzt werden.

Inhaltsverzeichnis

1	Einleitung	1	3	Nutzermakros	6
1.1	TL;DR	1	3.1	Automatisch Sprachmakros . .	6
1.2	Motivation	2	3.2	Sternvariante	7
1.3	Alternativen	2	3.3	Individuelle Sprachmakros . .	8
1.4	Sprachen	3	4	Interaktion mit anderen Paketen	8
			4.1	Babel/Polyglossia	8
			4.2	Csquotes	9
2	Anwendung	4	5	FAQ	9
2.1	Auswahl der Zielsprache . . .	4	6	Implementation	10
2.2	Optionen	5			

1 Einleitung

1.1 TL;DR

Angenommen Sie wollen von einem Text eine englische und eine deutsche Version erhalten, wobei Sie ein gemeinsames Quelldokument nutzen wollen..

1. Binden Sie `\usepackage[languages={en,de}]{langselect}` ein

¹. matthias.werner@informatik.tu-chemnitz.de

2. Nutzen Sie im Quelldokument `\lende{⟨Texten⟩}{⟨Textde⟩}`. `⟨Texten⟩` wird in der englischen Version erscheinen, während `Textde` in der deutschen Version erscheinen wird.
 - Text, der in beiden Versionen vorkommt, schreiben Sie außerhalb des Makros oder nutzen die Sternversion, um gemeinsame Daten im Fließtext nur einmal schreiben zu müssen, siehe Abschnitt 3.2.
3. Übersetzen Sie das Dokument mit

```
> latex "\def\olsTargetlanguage{en}\include{⟨document⟩}"
```

, um die englische Version zu erhalten, und mit

```
> latex "\def\olsTargetlanguage{de}\include{⟨document⟩}"
```

für die deutsche.
 - Es gibt noch einige andere Möglichkeiten, die Zielsprache zu bestimmen, u. a. über den Jobnamen und über `\DocumentMetadata`.

1.2 Motivation

Mitunter werden mehrere Sprachversionen des gleichen Dokuments benötigt. Im Anwendungsgebiet des `osglecture`-Bündel sind dies beispielsweise die Lehrskripte, Vorlesungsfolien und Handouts einer Lehrveranstaltung. Es fällt häufig schwer, mehrere parallel existierende Sprachversionen konsistent zu halten. Diese Aufgabe ist etwas einfacher, wenn alle Versionen in einem gemeinsamen Quelldokument enthalten sind.

Dieses Paket unterstützt die Generierung solcher Sprachversionen aus einem gemeinsamen \LaTeX -Quelldokument indem es Makros für Sprachvarianten und Mechanismen zur Auswahl der Zielsprache bereitstellt.

1.3 Alternativen

Es gibt eine Reihe von Alternativen zu `langselect`.

Adhoc-Makros Man kann relativ einfach adhoc Lösungen schaffen, wie z. B.

```
1 \newif\ifenglish
2 \englishtrue
3 \ifenglish
4     Welcome!
5 \else
6     ¡Bienvenido!
7 \fi
8
```

Tatsächlich folgt `langselect` grundsätzlich diesem Ansatz, automatisiert aber die Erstellung der Sprachmakros, vereinfacht die gemeinsame Nutzung von nichtsprachabhängigen

Elementen wie z. B. Formeln und ermöglicht die Auswahl der Sprachvariante über verschiedene Methoden.

multilang Das ausgeklügelte Paket `multilang`² von Richard GREWE erstellt Sprachversionen von gegebenen Makros, wobei die aktive Sprache direkt aus Babel oder Polyglossia übernommen wird. `langselect` hat ein ähnliches Ziel, aber ein anderes (aus Sicht des Paketautoren: praktischeres) Interface. Falls Sie aber den Ansatz von `langselect` nicht mögen, sollten Sie sich unbedingt `multilang` anschauen.

comment Victor EIJKHOUT hat das Paket `comment`³ geschrieben, das auf einfache Weise ermöglicht, nur bestimmte Abschnitte im Dokument auszugeben. Dies kann auch sehr gut für eine Sprachauswahl genutzt werden.

bicaption/translations/translator/xt_cpts Die Pakete `bicaption`⁴/`translations`⁵/`translator`⁶/`xt_cpts`⁷ adressieren Programmierer von \LaTeX -Paketen, um fixe Textelemente wie die Überschriften von Verzeichnissen zu internationalisieren. Für Dokumentautoren sind diese Pakete nur bedingt geeignet.

1.4 Sprachen

Das Konzept von `langselect` beinhaltet die Nutzung von Sprachen auf verschiedenen Ebenen mit unterschiedlicher Wirkung. Um Verwechslungen zu vermeiden, wollen wir hier die Begriffe eindeutig definieren.

1. Die Menge *möglicher* Zielsprachen des Dokuments. Ein Dokument, das `langselect` nutzt, enthält typischerweise Abschnitte in verschiedenen Zielsprachen. Die möglichen Zielsprachen werden über eine Paketoption gesetzt, siehe Abschnitt 2.2. Im weiteren Text dieser Dokumentation nennen wir die möglichen Zielsprachen *Auswahlsprachen*, da für die \LaTeX -Übersetzung eine dieser Sprachen ausgewählt wird.
2. Die tatsächliche bei einer \LaTeX -Übersetzung gebrauchte Zielsprache. Diese muss eine der Sprachen aus 1. sein. Typischerweise wird der Übersetzungsvorgang für jede der tatsächlichen Zielsprachen einzeln ausgeführt. Für die Auswahl der tatsächlichen Zielsprache gibt es mehrere Methoden, siehe Abschnitt 2.1.
3. Die Sprache, in der Textteile oder einzelne Wörter tatsächlich geschrieben sind. Dies ist das Konzept von Sprache, das Pakete wie Babel oder Polyglossia haben.⁸ Bitte beachten Sie, dass ein mehrsprachiges Dokument aus der Sicht von `langselect` ein monosprachiges Dokument aus der Sicht von Babel sein kann. Allerdings können durchaus noch weitere

2. on CTAN as `multilang`: <http://mirrors.ctan.org/macros/latex/contrib/multilang/>

3. on CTAN as `comment`: <http://mirrors.ctan.org/macros/latex/contrib/comment/>

4. on CTAN as `bicaption`: <http://mirrors.ctan.org/macros/latex/contrib/bicaption/>

5. on CTAN as `translations`: <http://mirrors.ctan.org/macros/latex/contrib/translations/>

6. on CTAN as `translator`: <http://mirrors.ctan.org/macros/latex/contrib/translator/>

7. on CTAN as `xt_cpts`: http://mirrors.ctan.org/macros/latex/contrib/xtcpts/xt_cpts/

8. Wobei auch in diesen Pakten „Sprache“ auch für mehrere unterschiedliche und z. T. voneinander unabhängige Konzepte wie Trennungsmuster, zu nutzende Fonts etc. steht

Sprachen hinzukommen: Beispielsweise kann ein Dokument mit den Auswahlssprachen Deutsch und Englisch im Text lateinische Wörter oder Sätze enthalten. Diese Auswahl wird über die üblichen Babelmechanismen wie `\selectlanguage`, `\foreignlanguage` oder `\text{<sprache>}` vorgenommen. Wenn Babel oder Polyglossia geladen wird, was auf Wunsch auch direkt über `langselect` erfolgen kann, wird als aktive Sprache die Zielsprache ausgewählt.

Sprachen werden in `langselect` über ISO 639-1-Codes (z. B. „en“, „fr“, „de“, ...) bezeichnet. Wenn für die Sprachbehandlung auf der Ebene von Babel/Polyglossia ein bestimmter Dialekt oder eine bestimmte Variante gewünscht ist, kann ein Mapping erfolgen.

2 Anwendung

Das Paket wird auf die übliche Weise geladen:

```
\usepackage[<options>]{langselect}
```

Da die Auswahl der Zielsprache sowohl über Optionen als auch auf anderem Weg erfolgen kann, werden hier zunächst diese verschiedenen Wege beschrieben. Weitere Optionen wie das Setzen der Auswahlssprachen werden in Abschnitt 2.2 besprochen.

2.1 Auswahl der Zielsprache

Um eine Zielsprache für einen \LaTeX -Übersetzungslauf zu bestimmen, gibt es verschiedene Möglichkeiten. Die Reihenfolge in der folgenden Liste spiegelt auch die Reihenfolge der Auswertung. Sobald damit eine Zielsprache ermittelt wird, wird eine weitere Auswertung abgebrochen.

1. Definition eines Makros `\olsTargetLanguage` vor Laden des Pakets. Damit ist es beispielsweise möglich, über

```
> latexmk -usepretex="\def\olsTargetLanguage{fr}" document.tex
```

die Zielsprache Französisch für den Übersetzungslauf festzulegen.

2. Setzen der Paketoption `targetlang`. Verschiedene Werte sind hier möglich:

```
targetlang = <ISO-Code>
```

Die Zielsprache wird direkt durch den Code angegeben.

```
targetlang = {job=<n>}
```

Die Zielsprache wird aus dem n . Element des Jobnamens bestimmt. Ein Element ist ein mit „-“ (Minus) abgetrennter Teil des Jobnamens, siehe die Dokumentation zum Paket `varsfromjobname`,⁹ das dafür geladen wird. Beispielsweise kann bei `targetlang = {job=2}` über

```
> latex -jobname doc-ru doc.tex
```

Russisch als Zielsprache eingestellt werden.

9. on CTAN as `varsfromjobname`: <http://mirrors.ctan.org/macros/latex/contrib/varsfromjobname/>

`targetlang = babel|polyglossia`

Für die Nutzung dieses Schlüsselwerts muss Babel oder Polyglossia *vor* `langselect` geladen sein. Die Zielsprache wird dann auf die durch das Sprachpaket eingestellte (Haupt-)Sprache gesetzt.¹⁰

`targetlang = meta`

Die in `\DocumentMetaData` angegebene Sprache als Zielsprache verwendet.

3. Wenn weder `\olsTargetLanguage` definiert noch die Option `targetlang` gesetzt wird, versucht `langselect` selbständig die Zielsprache zu bestimmen, indem es erst die Option `targetlang = babel` und dann (wenn kein Sprachpaket gefunden wurde) die Option `targetlang = meta` ausprobiert.
4. Wenn alles andere versagt, wird Englisch als Zielsprache gesetzt.

2.2 Optionen

Folgende Optionen stehen zur Verfügung:

`languages = {⟨Liste von Auswahl Sprachen⟩}` (erforderlich)

Gibt zwei oder drei Auswahl Sprachen an. Die Reihenfolge ist relevant für die automatische Generierung der Sprachmakros (siehe Abschnitt 3).

`targetlang = ⟨Sprache⟩|{job=⟨n⟩}|babel|polyglossia|meta`

Siehe Abschnitt 2.1

`prefix = {⟨prefix⟩}`

Voreinstellung: 1

Setzt einen Prefix für die Multisprachmakros. Es kann auch ein leerer Prefix gesetzt werden.

`auto = true|false`

Voreinstellung: true

Die eigentlichen Sprachmakros werden automatisch angelegt, siehe Abschnitt 3. Wird `auto = false` gesetzt, werden nur die Makros zur Erzeugung der Sprachmakros zur Verfügung gestellt, nicht die Sprachmakros selbst.

`trim = true|false`

Voreinstellung: true

Enternt Whitespaces am Anfang und Ende der Argumente der Sprachmakros.

`load babel = {⟨Babel Optionen⟩}`

Voreinstellung:

`load Polyglossia = {⟨Polyglossia Optionen⟩}`

Voreinstellung:

Lädt entweder Babel oder Polyglossia so, dass die Zielsprache von `langselect` als Hauptsprache gesetzt wird. Die hier übergebenen Optionen werden an das jeweilige Sprachpaket weitergeleitet. Wenn eine dieser Option genutzt wird, darf nicht `targetlang = babel` oder `targetlang = polyglossia` gesetzt sein und `langselect` muss vor dem jeweiligen Sprachpaket geladen werden.

10. Beide Werte bewirken exakt das Gleiche: Da Babel und Polyglossia die gleiche Schnittstelle anbieten, ist es egal, welches von beiden Paketen geladen ist. `targetlang = polyglossia` findet also auch Babel.

`map = {⟨Sprache⟩=⟨Variante⟩,...}`

Legt ein Mapping von Sprachen zu einer Sprachvariante fest. Wenn `langselect` mit anderen Paketen interagiert, wird dann die spezifische Sprachvariante statt des ISO-Codes weitergegeben.

Falls Sie beispielsweise Babel von `langselect` aus laden und Ihr Dokument explizit *Britisches Englisch* und *Schweizerdeutsch* nutzt, sichern Sie mit folgendem Code ab, dass Babel korrekte Optionen erhält.

```

1 \documentclass{article}
2 \usepackage[
3   languages={en,de},
4   load=babel,
5   map={en=british,de=nswissgerman}
6 ]
7 \begin{document}
8   \title{\lende{Hallo}{Grüezi}}
9 
```

`unified shorthands = true|false`

3 Nutzermakros

3.1 Automatisch Sprachmakros

Wenn Sie nicht die Option `auto = false` gesetzt haben, wird ein Multisprachmakro angelegt. Dessen Name wird durch die Liste der Auswahlssprachen festgelegt. Je nachdem, ob Sie zwei oder mehr Auswahlssprachen angegeben haben, ist der Name

- `\⟨Prefix⟩⟨Sprache1⟩⟨Sprache2⟩{⟨Text1⟩}{⟨Text2⟩}`

oder

- `\⟨Prefix⟩⟨Sprache1⟩⟨Sprache2⟩⟨Sprache3⟩{⟨Text1⟩}{⟨Text2⟩}{⟨Text3⟩}`

Beispielsweise für für diese Dokumentation das Paket mit folgenden Optionen geladen:

```

1 \usepackage[
2   languages={de,en},
3   targetlang={job=2}
4 ]{langselect}
```

Damit steht das Makro `\ldeen{⟨Text1⟩}{⟨Text1⟩}` zur Verfügung. Wenn jetzt die Zielsprache „de“ ausgewählt wird (hier über den Jobnamen, aber es gibt auch andere Möglichkeiten, vgl.

Abschnitt 2.1), wird nur $\langle Text_1 \rangle$ in das PDF-Dokument übernommen und $\langle Text_2 \rangle$ verworfen. Bei der Zielsprache „en“ ist es umgekehrt.

Achtung!

Die automatische Generierung der Sprachmakros kann manchmal zu Problemen führen. Beispielsweise ergibt Abchasisch (Code: „ab“) als Erst- und Griechisch (Code: „el“) als Zweisprache den automatischen Namen `\label`, was zu Fehlermeldungen führt. Nutzen Sie in solchen Fällen die im Abschnitt 3.3 beschriebenen Möglichkeiten.

3.2 Sternvariante

Ein häufiges Problem bei mehreren Sprachversionen ist die Konsistenz nichtsprachabhängiger Daten, insbesondere wenn diese Daten sich öfter ändern. (Denken Sie z. B. an Stundenpläne.)

Für solche Fälle stellt `langselect` eine Sternvariante der Sprachmakros bereit. In dieser Sternvariante können die Textargumente Platzhalter der Form „@ $\langle n \rangle$ “ enthalten, wobei $\langle n \rangle$ einen Wert zwischen 1 und 9 darstellt. Hinter dem Makro müssen dann weitere Argumente angegeben werden, und zwar so viele, wie es dem höchsten verwendeten $\langle n \rangle$ entspricht. Die Sternvariante ersetzt die Platzhalter mit den entsprechenden Argumenten.

Die Sternvariante der Sprachmakros ist „zerbrechlich“, bitte verwenden Sie sie nicht in einem beweglichen Kontext, z. B. innerhalb eines `\section`-Arguments.

```

1  \documentclass{article}
2  \usepackage[languages={en,fr},
3  targetlang=fr]{langselect}
4
5  \begin{document}
6  \lenfr*{
7    The formula @1 generally refers to the Pythagorean theorem.
8  }{
9    La formule @1 fait généralement référence au théorème de
10   Pythagore.
11   }{ $a^2+b^2=c^2$ }
12
13  \lenfr*{
14    Calculate the length of the third side if one side is @1 units long
15    and the hypotenuse is @2 units long.
16  }{
17    Calculez la longueur du troisième
18    côté si un côté mesure @1 unités et l'hypoténuse @2 unités.
19  }{ $\$4$ }{ $\$5$ }
20  \end{document}

```

3.3 Individuelle Sprachmakros

Wenn Ihnen die generierten Namen der Sprachmakros nicht gefallen oder diese sogar zu Kollisionen führen, können Sie diese ändern. Am einfachsten ist es, die Paketoption `prefix` zu ändern, so dass beispielsweise mit `prefix = mv` das Sprachmakro für die Deutsch/Englisch-Kombination `\mvdeen` heisst.

Wenn dies nicht ausreichend ist, geben Ihnen die folgenden zwei Makros die vollständige Kontrolle über die Namensbildung.

```
\olsMakeBilingualMacro[⟨Prefix⟩]{⟨Sprache1⟩}{⟨Sprache2⟩}[⟨Basisname⟩]
```

```
\olsMakeTrilingualMacro[⟨Prefix⟩]{⟨Sprache1⟩}{⟨Sprache2⟩}{⟨Sprache3⟩}[⟨Basisname⟩]
```

Diese Makros legen ein bilinguals bzw. trilinguales Sprachmakro für die angegebenen Sprachen an. Wenn `⟨Basisname⟩` angegeben wird, ist der Name des Sprachmakros `\⟨Prefix⟩⟨Basisname⟩`. Wenn das `⟨Prefix⟩`-Argument benutzt wird, bestimmt dieses den Prefix, sonst gelten die Paketoptionen.

4 Interaktion mit anderen Paketen

4.1 Babel/Polyglossia

Das Paket `langselect` ist zunächst einmal unabhängig von Sprachpaketen wie Babel oder Polyglossia. Es kann ohne oder neben diesen Paketen eingesetzt werden. Wenn man sich jedoch die Mühe eines mehrsprachigen Dokuments macht, will man in der Regel auch, dass für die jeweilige Sprache die richtigen Trennungsregeln, Fonteneinstellungen, Datumsformate etc. benutzt werden.¹¹

Bitte beachten Sie, dass, solange Sie nicht *innerhalb* eines Textes eine andere Sprache (vielleicht ja die Sprache der anderen Sprachversion) nutzen, ihr Dokument aus der Sicht von Babel ein *einsprachiges* Dokument ist. Es besteht die Gefahr, dass Sie Babel auf die gewohnte Weise mit den verschiedenen Sprachen ihrer Sprachversion aufrufen, aber zur Bestimmung der Zielsprache nicht die Option `targetlang = babel` nutzen. Als Ergebnis stimmt u.U. die Zielsprache nicht mit Babels aktiver Sprache überein, und können Sie inkorrekte Trennungen und ähnliche Fehler erhalten.

Um solche Probleme zu vermeiden, bieten `langselect` verschiedene Möglichkeiten:

1. Wie oben erwähnt kann man die Paketoption `targetlang = babel` nutzen. Dann wird die Zielsprache durch die Hauptsprache von Babel bestimmt. Allerdings muss man dann auf die Möglichkeit verzichten, die Zielsprache „von außen“ (also mit `\de` oder über den Jobnamen) zu steuern.
2. Umgekehrt lädt `langselect` Babel oder Polyglossia mit Hilfe der Optionen `load babel/` `load polyglossia` landen und setzt die Zielsprache als Hauptsprache. Weitere Optionen an die Sprachpakete werden als Optionsargument übergeben.

¹¹ Im Folgenden werde ich mich ausschließlich auf Babel beziehen. Das Gesagte gilt jedoch entsprechend auch für Polyglossia.

3. Schließlich kann mit dem Hook `langselect/language` beliebiger Code vor Ausgabe der Argumente des Sprachmakros eingefügt werden, z. B. kann hier `\foreignlanguage` genutzt werden. Der Hook hat als Parameter die Zielsprache.

4.2 Csquotes

Unterschiedliche Sprachen haben unterschiedliche Konventionen zur Setzung von Satzzeichen. Besonders stört das bei Anführungszeichen, da diese der Zielsprache folgen sollen, aber durchaus auch außerhalb der Sprachmakros oder in Argumenten der Sternvariante stehen könne. Das Paket `csquote`¹² bietet dafür eine Lösung, indem es `\enquote` und andere Makros anbietet, die der Zitatkonvention der aktuellen Sprache folgen.

Zur Erhöhung der Bequemlichkeit unterstützt `langselect` die Nutzung korrekter Anführungszeichen mit sprachunabhängigen Shorthands: Wenn die Option `unified shorthands` wahr ist, wird `csquotes` geladen und die Shorthands

``` und

`'` auf `\openautoquote` bzw. `\closeautoquote` gesetzt.

Damit werden in Sprachen wie Deutsch, wo diese Shorthands

## 5 FAQ

### • Ist `langselect` noch experimentell?

Auf jeden Fall. Während ist beim Autoren zwar im täglichen Einsatz ist, ist nicht auszuschließen, dass es unerwünschte Wechselwirkungen zu anderen Paketen gibt. Die Versionsnummer zeigt dies an.

### • Wofür steht das „o“ im `\ols`-Prefix??

Das Paket gehört zu Bundle `*osglecture`, von dort ist das „o“ geerbt.

### • Gibt es Anwendungsbeispiele, die über ein Toy-Hello-World hinausgehen?

Ja, einige. Gucken Sie doch mal zum Beispiel in den Quelltext dieser Dokumentation.

### • Ich habe ein `@1` in der Ausgabe, wo doch eine Argumentersetzung stattfinden sollte.

Das hat in der Regel eine der beiden folgenden Ursache:

1. Sie haben den Stern im Sprachmakro vergessen.
2. Sie haben komische Dinge mit Catcodes angestellt.

### • Kann ich in der Sternversion einen `@n`-Parameter auslassen?

Nein. Außerdem wäre dann das Argument ja auch offensichtlich nicht mehr gemeinsam.

---

12. on CTAN as `csquote`: <http://mirrors.ctan.org/macros/latex/contrib/csquote/>

- Kann ich  $@n$ -Parameter in gemischter Reihenfolge oder mehr als einmal verwenden?

Ja.

- Ich möchte @ in einer Sternvariante des Sprachmacros verwenden.

Dann machen Sie es.

- Aber ich brauche @42.

Nutzen Sie @{}42

- Ich nutze Russisch, aber der Font ist nicht richtig.

Das ist keine Problem von langselect (das kümmert sich gar nicht um den Font), sondern wahrscheinlich von Babel, `\fontenc`, `\fontspec`, etc.

- Meine Frage erscheint hier nicht.

## 6 Implementation

```

1 \NeedsTeXFormat{LaTeX2e}[2022/06/01]
2 \def\packagename{langselect}
3 \def\packageversion{2025/09/04 v0.2.1}
4 \ProvidesPackage{\packagename}[\packageversion\space support for different
5 languages versions from common source]
6 \ExplSyntaxOn

```

`\olsIsoTags` enthält alle gültigen ISO 639-1-Codetags.

```

7 \seq_const_from_clist:cn{\olsIsoTags}{aa,ab,ae,af,ak,am,an,ar,as,av,ay,az,ba,be,
8 bg,bh,bi,bm,bn,bo,br,bs,ca,ce,ch,co,cr,cs,cu,cv,cy,da,de,dv,dz,ee,el,en,eo,es,
9 et,eu,fa,ff,fi,fj,fo,fr,fy,ga,gd,gl,gn,gu,gv,ha,he,hi,ho,hr,ht,hu,hy,hz,ia,id,
10 ie,ig,ii,ik,io,is,it,iu,ja,jv,ka,kg,ki,kj,kl,km,kn,ko,kr,ks,ku,kv,kw,ky,la,
11 lb,lg,li,ln,lo,lt,lu,lv,mg,mh,mi,mk,ml,mn,mr,ms,mt,my,na,nb,nd,ne,ng,nl,nn,no,
12 nr,nv,ny,oc,oj,om,or,os,pa,pi,pl,ps,pt,qu,rm,rn,ro,ru,rw,sa,sc,sd,se,sg,si,sk,
13 sl,sm,sn,so,sq,sr,ss,st,su,sv,sw,ta,te,tg,th,ti,tk,tl,tn,to,tr,ts,tt,tw,ty,ug,
14 uk,ur,uz,ve,vi,vo,wa,wo,xh,yi,yo,za,zh,zu}
15
16 \newcommand\IfTagIsValidF[2]{%
17 \seq_if_in:ceF{\olsIsoTags}{#1}{#2}
18 }
19 \RequirePackage{varsfromjobname}[2025/08/03]

```

Das Ergebnisse von `\varsfromjobname` und `\str_range:Nnn` haben Catcode 12, aber Iso-Tags und Optionen haben Catcode 11. Wir vereinheitlichen durch einen Wrapper auf Catcode 11.

```

20 \newcommand\olsChangeCatEleven[1]{
21 \def\tmpa{#1}
22 \edef\tmpb{
23 \noexpand\scantokens{
24 \noexpand\edef\noexpand\olsCatElven{\tmpa}
25 }
26 }
27 \tmpb
28 }

```

Bei Bedarf bilden wir Spachcode auf Sprachvarianten ab. Bei Interaktionen mit externen Code wir dann die Sprachvariante weitergegeben.

```

29 \cs_new:Npn \ols_use_map:n #1 {
30 \cs_if_exist_use:cF { olslangmap / #1 }{ #1 }
31 }

```

Wir definieren einige Optionen. Alle außer `language` sind optional. Einige Ergebnisse werden wir in Variablen speichern, die wir zuerst anlegen.

```

32 \bool_new:N\ols_load_babel
33 \bool_new:N\ols_load_polyglossia
34 \DeclareKeys{
35 languages.clist_gset:N = \ols_langs,
36 languages.usage=load,
37 targetlang.code = {
38 \cs_if_exist:NF\olsTargetLanguage{
39 \str_if_eq:eeT{\str_range:Nnn{#1}{1}{3}}{job}{%
40 \olsChangeCatEleven{\getfromjobname{\str_range:Nnn{#1}{-1}{-1}}}
41 \xdef\olsTargetLanguage{\olsCatElven}
42 }
43 \bool_if:nTF{
44 \str_if_eq_p:nn{#1}{babel} || \str_if_eq_p:nn{#1}{polyglossia}
45 }{
46 \str_if_empty:eF{\BCPdata{main.language}}{
47 \edef\olsTargetLanguage{\BCPdata{main.language}}
48 }
49 }
50 \str_if_eq:nnTF{#1}{meta}{
51 \IfDocumentMetadataTF{
52 \edef\tmpa{\GetDocumentProperty{document/lang}}
53 \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
54 \olsChangeCatEleven{\tmpb}
55 \edef\olsTargetLanguage{\olsCatElven}
56 }{

```

## 6 Implementation

```
57 \PackageWarningNoLine{\packagename}{No~meta~data~provided}
58 }
59 }
60 }
61 },
62 targetlang.usage=load,
63 prefix.store=\ols_sprefix,
64 prefix.initial:n=l,
65 prefix.usage=load,
66 auto.bool_set:N=\ols_generate,
67 auto.usage=load,
68 auto.initial:n=true,
69 trim.bool_set:N=\ols_trim,
70 trim.usage=load,
71 trim.initial:n=true,
72 map.code:n = {
73 \prop_new:N \ols_lang_pl
74 \prop_clear:N \ols_lang_pl
75 \prop_set_from_keyval:Nn \ols_lang_pl {#1}
```

Da Property-Listen nicht expandierbar sind, setzen wir die Einträge auf gewöhnlich Makros um.

```
76 \prop_map_inline:Nn \ols_lang_pl
77 { \cs_gset:cpn { olslangmap / \tl_to_str:n {##1} } {##2} }
78 },
79 load~babel .code:n = {
80 \bool_set_true:N \ols_load_babel
81 \PassOptionsToPackage{#1}{babel}
82 },
83 load~polyglossia .code:n = {
84 \bool_set_true:N \ols_load_polyglossia
85 \PassOptionsToPackage{#1}{polyglossia}
86 },
87 load.usage = load,
88 unified~shorthands.bool_set:N=\ols_shorthands,
89 unified~shorthands.initial:n=true,
90 unified~shorthands.usage=load
91 }%
92 \ProcessKeyOptions\relax
```

Wenn über `\olsTargetLanguage` oder die `targetlang` keine Zielsprache zugewiesen wurde, untersuchen wir noch einmal, ob über Sprachpakete oder `\DocumentMetadata` eine Sprache identifiziert werden kann.

## 6 Implementation

```
93 \cs_if_exist:NF \olsTargetLanguage {
94 \str_if_eq:eeF{\BCPdata{main.language}}{\edef\olsTargetLanguage{\BCPdata{main.language}}}
95 }
96 \cs_if_exist:NF \olsTargetLanguage {
97 \IfDocumentMetadataT{
98 \edef\tmpa{\GetDocumentProperty{document/lang}}
99 \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
100 \olsChangeCatEleven{\tmpb}
101 \edef\olsTargetLanguage{\olsCatElven}
102 }
103 }
104 \cs_if_exist:NF \olsTargetLanguage {
105 \PackageWarningNoLine{\packagename}{Can't~identify~any~target~language.
106 \MessageBreak Falling~back~to~'en'}
107 \def\olsTargetLanguage{en}
108 }
```

Wir führen verschiedene Checks durch. Als erstes prüfen wir, ob eine valide Zielsprache ermittelt werden konnte.

```
109 \IfTagIsValidF{\olsTargetLanguage}{
110 \PackageWarningNoLine{\packagename}{Do~not~recognize~language~'\olsTargetLanguage'.
111 \MessageBreak Take~'en'~as~replacement}
112 \def\olsTargetLanguage{en}
113 }
```

Sind mindestens zwei Auswahlssprachen angegeben?

```
114 \int_compare:nNtT{\clist_count:N\ols_langs} < {2}{
115 \PackageError{\packagename}{No~sufficient~number~of~selectable
116 \MessageBreak~languages~provided}{
117 You~have~to~provide~a~list~of~two~or~three~\MessageBreak
118 languages~via~'language'~option.}
119 \aftergroup\endinput
120 }
```

Im Moment kann langselect nicht mehr als drei Auswahlssprachen verarbeiten. Wir warnen daher, wenn mehr als drei Sprachen angegeben werden.

```
121 \int_compare:nNtT{\clist_count:N\ols_langs} > {3}{
122 \PackageWarningNoLine{\packagename}{Too~many~selectable~languages~provided.
123 \MessageBreak I~will~ignore~the~superfluous~languages}
124 }
```

Als nächstes überprüfen wir, ob alle Auswahlssprachen gültige ISO 639 1 Codes sind.

## 6 Implementation

```
125 \clist_map_inline:Nn\ols_langs {
126 \IfTagIsValidF{#1}{
127 \PackageError{\packagename}{Couldn't~resolve~selectable~language~'#1'}{
128 Use~valid~ISO-639-1~code~in~option~'languages'.}
129 \aftergroup\endinput
130 }
131 }
```

Die Zielsprache sollte in der Liste von Auswahlssprachen enthalten sein.

```
132 \clist_if_in:NVF\ols_langs{\olsTargetLanguage}{
133 \PackageError{\packagename}{Target~language~'\olsTargetLanguage'~is~not~\MessageBreak
134 in~the~list~of~selectable~languages}{
135 Check~option~'language'~and~'targetlang'.}
136 }
```

Babel und Polyglossia können nicht beiden geladen werden.

```
137 \bool_if:nT{
138 \ols_load_babel && \ols_load_polyglossia
139 }{
140 \bool_set_false:N \ols_load_babel
141 \bool_set_false:N \ols_load_polyglossia
142 \PackageError{\packagename}{The~options~'load~babel'~and~'load~polyglossia'
143 \MessageBreak are~mutual~exclusive,~you~can't~use~both}{Use~one~load~option,~only.}
144 }
145 \bool_if:nT{
146 (\cs_if_exist_p:c{ver@babel.sty} || \cs_if_exist_p:c{ver@polyglossia.sty})
147 &&
148 (\ols_load_babel && \ols_load_polyglossia)
149 }{
150 \bool_set_false:N \ols_load_babel
151 \bool_set_false:N \ols_load_polyglossia
152 \PackageWarningNoLine{\packagename}{Can't~load~babel/polyglossia~for
153 ~you,since~\MessageBreak
154 a~language~package~is~already~loaded.\MessageBreak
155 I'll~ignore~that~option}
156 }
```

`\ols_mapped_target_lang` ist die Bezeichnung der Zielsprache, wie sie ggf. an externen Code weitergegeben wird, d.h. `\DocumentMetadata`, Sprachpakete und Hooks.

```
157 \edef\ols_mapped_target_lang{\ols_use_map:n{\olsTargetLanguage}}
```

Wir korrigieren noch ggf. die über `\DocumentMetadata` gesetzten Metadaten auf die der Zielsprache, so dass beispielsweise Babel keine falsche Hauptsprache ableitet.

## 6 Implementation

```

158 \IfDocumentMetadataT{
159 \edef\olsDocLang{\exp_args:Ne\str_range:nnn{\GetDocumentProperty{document/lang}}{1}{2}}
160 \bool_if:nF{
161 \str_if_eq_p:ee{\olsDocLang}{\olsTargetLanguage}
162 }{
163 \PackageWarningNoLine{\packagename}{Target~language~'\olsTargetLanguage'~
164 doesn't~comply~with\MessageBreak \c_backslash_str
165 DocumentMetadata~'\GetDocumentProperty{document/lang}'.\MessageBreak
166 I'll~try~to~overwrite~the~meta~data}
167 \DocumentMetadata{lang=\ols_mapped_target_lang}
168 }
169 }

```

Da der Name des Sprachmakros nicht bekannt ist, sind Patchen oder generische Hooks komplizierter. Wir stellen deshalb eigene Hooks zur Verfügung.

```

170 \NewHookWithArguments{langselect/language}{1}
171 \NewHookWithArguments{langselect/argument}{2}

```

Ein Boolean zur Markierung, ob mehr als zwei Auswahlssprachen angegeben wurden.

```

172 \bool_new:c{ols_trilang}
173 \int_compare:nNnTF{\clist_count:N{\ols_langs}} > {2}{
174 \bool_set_true:c{ols_trilang}
175 }{
176 \bool_set_false:c{ols_trilang}
177 }
178 \def\olsFrstLanguage{\clist_item:Nn \ols_langs{1}}
179 \def\olsScndLanguage{\clist_item:Nn \ols_langs{2}}
180 \bool_if:cT{ols_trilang}{
181 \def\olsThrdLanguage{\clist_item:Nn \ols_langs{3}}
182 }
183 \bool_if:NTF\ols_trim{
184 \let\ols_trim:n=\tl_trim_spaces:n
185 }{
186 \let\ols_trim:n=\relax
187 }

```

Die Makros `\olsMakeBilingualMacro` bzw. `\olsMakeTrilingualMacro` dienen zur Generierung der eigentlichen Sprachmakros.

```

188 % #1: prefix, #2,#3: selectable languages, #4: base name
189 \NewDocumentCommand{\olsMakeBilingualMacro}{0{\ols_sprefix} m m o}{
190 \str_if_eq:eeT{\olsTargetLanguage}{#2} {
191 \IfValueTF{#4}{
192 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#4}{s +m +m}{

```

## 6 Implementation

```

193 \UseHook{langselect/language}{\ols_mapped_target_lang}
194 \IfBooleanTF{##1}{\ols_trim:n{\olsProcessArgs{##2}}}{\ols_trim:n{##2}}
195 }
196 }{
197 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3}{s +m +m}{
198 \UseHook{langselect/language}{\ols_mapped_target_lang}
199 \IfBooleanTF{##1}{\ols_trim:n{\olsProcessArgs{##2}}}{\ols_trim:n{##2}}
200 }
201 }
202 }
203 \str_if_eq:eeT{\olsTargetLanguage}{#3}{
204 \IfValueTF{#4}{
205 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#4}{s +m +m}{
206 \UseHook{langselect/language}{\ols_mapped_target_lang}
207 \IfBooleanTF{##1}{\ols_trim:n{\olsProcessArgs{##3}}}{\ols_trim:n{##3}}
208 }
209 }{
210 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3}{s +m +m}{
211 \UseHook{langselect/language}{\ols_mapped_target_lang}
212 \IfBooleanTF{##1}{\ols_trim:n{\olsProcessArgs{##3}}}{\ols_trim:n{##3}}
213 }
214 }
215 }
216 }

```

Nun das Gleiche nochmal für *drei* Sprachen.

```

217 % #1: prefix, #2...#4: selectable languages, #5: base name
218 \NewDocumentCommand{\olsMakeTrilingualMacro}{0{\ols_sprefix} m m m o}{
219 \str_if_eq:eeT{\olsTargetLanguage}{#2} {
220 \IfValueTF{#4}{
221 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
222 \UseHook{langselect/language}{\ols_mapped_target_lang}
223 \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##2}}
224 }
225 }{
226 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
227 \UseHook{langselect/language}{\ols_mapped_target_lang}
228 \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##2}}
229 }
230 }
231 }
232 \str_if_eq:eeT{\olsTargetLanguage}{#3} {
233 \IfValueTF{#4}{
234 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{

```



## 6 Implementation

```

235 \UseHook{langselect/language}{\ols_mapped_target_lang}
236 \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##3}}
237 }
238 }{
239 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
240 \UseHook{langselect/language}{\ols_mapped_target_lang}
241 \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##3}}
242 }
243 }
244 }
245 \str_if_eq:eeT{\olsTargetLanguage}{#4} {
246 \IfValueTF{#4}{
247 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#5}{s +m +m +m}{
248 \UseHook{langselect/language}{\ols_mapped_target_lang}
249 \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##4}}
250 }
251 }{
252 \ExpandArgs{c}\NewExpandableDocumentCommand{#1#2#3#4}{s +m +m +m}{
253 \UseHook{langselect/language}{\ols_mapped_target_lang}
254 \IfBooleanTF{##1}{\ols_trim:{\olsProcessArgs{##2}}}{\ols_trim:{##4}}
255 }
256 }
257 }
258 }

```

Falls die automatische Erzeugung nicht deaktiviert ist, werden die konkreten Sprachmakros angelegt. Der Name sind  $\langle \text{Prefix} \rangle \langle \text{Sprache}_1 \rangle \langle \text{Sprache}_2 \rangle$  bzw.  $\langle \text{Prefix} \rangle \langle \text{Sprache}_1 \rangle \langle \text{Sprache}_2 \rangle \langle \text{Sprache}_2 \rangle$ .

```

259 \bool_if:cTF{ols_generate}{
260 \bool_if:cTF{ols_trilang}{
261 \olsMakeTrilingualMacro{\olsFrstLanguage}{\olsScndLanguage}{\olsThrdLanguage}
262 }{
263 \olsMakeBilingualMacro{\olsFrstLanguage}{\olsScndLanguage}
264 }
265 }{
266 \PackageInfoNoLine{\packagename}{No~language~macros~generated.\MessageBreak
267 Use~\string\olsMakeBilingualMacro~or~\string\olsMakeTrilingualMacro~to~generate.}
268 }

```

Für die Sternvarianten der Sprachmakros legen wir uns ein Makro an, dass die folgenden Argumente verarbeitet.

```

269
270 \seq_new:N \ols_args_seq
271 \int_new:N \ols_num_of_args

```

```

272 \tl_new:N \ols_format_str_tl
273

```

Da das Platzhalterzeichen @ im folgendem Code als Literal auftritt, muss es den gleichen Catcode wie im Dokument haben.

```

274 \makeatother
275 \NewDocumentCommand \olsProcessArgs { +m } {
276 \tl_gset:Nn \ols_format_str_tl{#1}
277 \int_zero:N \ols_num_of_args

```

Suche den höchsten Wert  $n$ , für den @ $n$  im Formatstring vorkommt.

```

278 \int_step_inline:nnnn { 9 } { -1 } { 1 }
279 {
280 \regex_if_match:nnT {@##1}{ #1 }
281 {
282 \int_set:Nn \ols_num_of_args { ##1 } \prg_break:
283 }
284 }

```

Wenn kein Platzhalterzeichen gefunden wurde, gib den Text in Originalform aus. Ansonsten wird das Ersetzungsmakro gerufen.

```

285 \int_compare:nNnTF {\ols_num_of_args} = { 0 }
286 { #1 }{
287 \seq_clear:N \ols_args_seq
288 \ols_collect_replace:n \ols_num_of_args
289 }
290 }

```

`\ols_collect_replace` erhält *ein* Argument übergeben, nämlich die Anzahl der verbleibenden Ersetzungstexte. Es konsumiert aber *zwei* Argumente. Dieses zweite holt es sich aus dem nach dem Makro liegenden Tokenstrom, wo es den jeweilig nächsten Ersetzungstext findet. Dieser wird in einer Tokensequent gespeichert und `\ols_collect_replace` erneut aufgerufen.

```

291 \cs_new_protected:Npn \ols_collect_replace:n #1 #2{
292 \IfHookEmptyTF{langselect/argument}{
293 \seq_put_right:Nn \ols_args_seq { #2 }
294 }{
295 \seq_put_right:Nn \ols_args_seq {
296 \UseHook{langselect/argument}{\ols_mapped_target_lang}{#2}
297 }
298 }
299 \int_compare:nNnTF { #1 } = { 1 }
300 {

```

Wenn alle Ersetzungstexte eingelesen wurden, werden sie nacheinander auf die Platzhalterausdrücke angewendet.

```

301 \int_step_inline:nn { \ols_num_of_args }
302 {
303 \tl_analysis_log:n{@}
304 \tl_analysis_log:n{\ols_format_str_tl}
305 \tl_replace_all:Nnn \ols_format_str_tl
306 { @##1 }
307 { \seq_item:Nn \ols_args_seq { ##1 } }
308 }
309 \ols_format_str_tl
310 }{
311 \ols_collect_replace:n { \int_eval:n { #1 - 1 } }
312 }
313 }
314 \makeatletter

```

Wenn gewünscht, wird Babel oder Polyglossia geladen und die Zielsprache als Hauptsprache gesetzt.

```

315 \bool_if:NT \ols_load_babel{
316 \RequirePackage[main=\ols_mapped_target_lang, provide*=*]{babel}
317 \bool_if:nT{
318 \ols_shorthands && !\str_if_p:ee{olsTargetLanguage}{de}{
319 \languageshorthands{ngerman}
320 }
321 }
322 }
323 \bool_if:NT\ols_load_polyglossia{
324 \RequirePackage{polyglossia}
325 }

```

Unabhängig davon, wann ein Sprachpaket geladen wurde, wird versucht, die Zielsprache zu aktivieren.

```

326 \cs_if_exist:cT{ver@babel.sty}{
327 \AtBeginDocument{
328 \selectlanguage{\ols_mapped_target_lang}
329 }
330 }
331 \cs_if_exist:cT{ver@polyglossia.sty}{
332 \bool_if:NTF\ols_shorthands{
333 \setdefaultlanguage[babelshorthands=true]{\ols_mapped_target_lang}
334 }{
335 \setdefaultlanguage{\ols_mapped_target_lang}

```

```

336 }
337 }
338 \bool_if:NT\ols_shorthands {
339 \RequirePackage[autostyle=true]{csquotes}
340 \AtBeginDocument{
341 \cs_if_exist:cT{usesshorthands}{
342 \usesshorthands*{"}
343 \defineshorthand{"`"}{\openautoquote}
344 \defineshorthand{"'"}{\closeautoquote}
345 \defineshorthand{"{/"}{\textormath{\bbl@allowhyphens
346 \discretionary{/}{}{/}\bbl@allowhyphens}{}}
347 }
348 }
349 }

```