

DAS TAGPAX-PAKET

vo.8.6 2026-10-22

Semantischer Import getaggtter PDF-Dokumente

Matthias WERNER

TAGPAX ist ein experimentelles Lua¹TeX-Paket zum Zusammenstellen vollständig getaggtter PDF-Dokumente. Es übernimmt Seiten, logische Dokumentstruktur und Navigation in ein neues Masterdokument.

Das Paket gehört zum **oslecture**-Bundle, kann aber auch eigenständig genutzt werden.

Inhaltsverzeichnis

1 Ziel und Anwendungsbereich	1	3.3.4 Import planning: tagpax-import	6
2 Nutzung	2	3.3.5 LuaTeX importer: tagpax-luatex	7
2.1 Schnellstart	2	3.3.6 Native importer: tagpax-native	7
2.2 Optionale PDFPAGES-Kompatibilität	2	3.3.7 Backend prototype: tagpax-backend	7
2.3 Konfiguration	3	3.4 Canonical intermediate repre- sentation	7
2.4 Befehlsübersicht	3	3.5 Backend	8
3 Developer manual	4	3.6 Implementation	8
3.1 Entwicklungsstand	4	3.6.1 tagpax.sty	8
3.2 Architektur	5	3.6.2 tagpdf-bridge	12
3.3 Lua API	6	3.6.3 tagpax.lua	26
3.3.1 Inspector: tagpax- inspect	6		
3.3.2 IR module: tagpax-ir	6		
3.3.3 Validation: tagpax- validate	6		
		Index	44

1 Ziel und Anwendungsbereich

Es gibt häufig den Fall, dass fertige PDF-Dokumente in andere eingebaut werden, beispielsweise bei Proceedings und Sammelbänden. Dafür hat sich Andreas MATTHIAS' umfangreiches Paket **pdfpages**¹ bewährt. Um aus den eingebundenen Dokumenten PDF-Annotationen zu

1. auf CTAN als pdfpages: <https://ctan.org/tex-archive/macros/latex/contrib/pdfpages>

übernehmen, beispielsweise zur Erstellung von gemeinsamen Inhaltsverzeichnissen, kann zusätzlich Ulrike FISCHERS `newpax`² oder dessen auf Java basierendes Vorgängerprojekt `pax`³ von Heiko OBERDIEK genutzt werden.

Jedoch gibt es aus der Perspektive der Barrierefreiheit ein Problem: Auch wenn die eingebundenen Dokumente (über `\MetaDocumentdata`) korrekt getaggt sind, gehen diese Informationen beim Einbinden verloren. Dies war der Anstoß für die Schaffung von `TAGPAX`, das die Möglichkeit gibt, Tagging-Informationen in das Gesamtdokument zu übernehmen.

Achtung

Dieses Paket ist **kein** Ersatz für die oben genannten Pakete `pdfpages` und `pax/newpax`. Es ist (zumindest derzeit) nur auf den Anwendungsfall der Inkludierung von vollständigen, korrekt getaggt PDF-Dokumenten in der Originalreihenfolge der Seiten ausgelegt. Die vielfältigen Möglichkeiten zu weiteren Manipulation sind in `TAGPAX` nicht möglich.

Beim Einbinden eines Subdokuments wird seine äußere Document-Struktur entfernt; ihre Kinder werden im Master unter einem neuen Part eingehängt. Vorhandene MCIDs bleiben dabei erhalten.

2 Nutzung

2.1 Schnellstart

Aktivieren Sie PDF-Management und automatisches Tagging vor der Dokumentklasse mit `\DocumentMetadata{tagging=on}` sowohl in den Subdokumenten, als auch im Masterdokument.

```
1 \DocumentMetadata{tagging=on}
2 \documentclass{book}
3 \usepackage{tagpax}
```

Um ein getaggt Dokument einzubinden, extrahieren Sie zunächst die semantische Zwischenrepräsentation mit `\tagpaxextract` und importieren Sie anschließend das vollständige Dokument mit `\tagpaxinclude`.

```
1 \tagpaxextract[paper.tagpax]{paper.pdf}
2 \tagpaxinclude[ir=paper.tagpax]{paper.pdf}
```

Der Importer erzeugt pro Quellseite ein frisches Form-XObject, setzt `/StructParents` vor dem Schreiben des Forms und rekonstruiert die Seitenstream-MCRs unter einem neuen Part. Explizite verschachtelte `/Stm-Streams` sind im gegenwärtigen MVP noch nicht importierbar.

2. auf CTAN als `newpax`: <https://ctan.org/tex-archive/macros/latex/contrib/newpax>

3. auf CTAN als `pax`: <https://ctan.org/tex-archive/macros/latex/contrib/pax>

2.2 Optionale PDFPAGES-Kompatibilität

Für eine schnelle Migration kann die Paketoption `pdfpages = true` die vertraute `\includepdf`-Syntax bereitstellen. Unterstützt werden nur der lineare Vollimport mit `pages = {-}` und `pagecommand`. Die Option lädt `pdfpages` nicht; die eigentliche Form-Erzeugung übernimmt der native Importer.

```

1 \usepackage[pdfpages]{tagpax}
2 \includepdf[
3   pages=-,
4   pagecommand={\thispagestyle{empty}}
5 ]{paper.pdf}

```

Hinweis

Aktivieren Sie diese Option nicht zusammen mit dem echten `pdfpages`-Paket, das könnte zu Problemen führen. Wenn `tagpax pdfpages` erkennt, deaktiviert es seinen Kompatibilitätsbefehl und gibt eine Warnung aus. Verwenden Sie in einem solchen Dokument für semantische Importe ausschließlich `\tagpaxinclude`. Optionen wie `nup`, `booklet`, `signature`, beliebige Seitenauswahlen und wiederholte Seiten werden abgelehnt, weil (zumindest derzeit) vollständige lineare Beiträge vorausgesetzt werden.

2.3 Konfiguration

Um die Strukturen korrekt in das aufnehmende Dokument einzupassen, kann das Tagging konfiguriert werden.

```

1 \tagpaxsetup{
2   heading-map = {H2=chapter,H3=section,H4=subsection},
3   toc-depth = H4,
4   bookmark-depth = H5
5 }

```

Die Option Der Schlüssel `heading-map` ordnet PDF-Überschriftenrollen den Ebenen des LaTeX-Inhaltsverzeichnisses zu. `toc-depth` und `bookmark-depth` legen die maximale Tiefe für sichtbares Inhaltsverzeichnis und PDF-Lesezeichen getrennt fest. Die Überschriftenrollen im importierten Strukturbaum werden nicht verändert.

2.4 Befehlsübersicht

Insgesamt stellt das Paket `TAGPAX` die folgende Macros zur Verfügung:

`\tagpaxextract[⟨Ausgabedatei⟩]{⟨PDF-Datei⟩}`

Extrahiert die kanonische Zwischenrepräsentation aus `⟨PDF-Datei⟩`. Ohne explizite `⟨Ausgabedatei⟩` wird der Dateiname aus der PDF-Datei abgeleitet.

`\tagpaxinclude[⟨ir=metadata file⟩]{⟨PDF-Datei⟩}`

Importiert PDF-Datei mit dem nativen linearen Importer. Mit `ir` kann eine bereits erzeugte Zwischenrepräsentation angegeben werden. Mit dem Schlüssel `layout` kann zusätzlich getaggte Imposition angefordert werden, z. B. `layout=4 on 1` für vier gleichwertige Quellseiten pro Zielblatt in einem 2×2-Raster (unterstützte Werte: 2, 3, 4 und 6 on 1). Bei 2 on 1 und 3 on 1 kann ein Suffix `|ruled` angehängt werden (`layout=2 on 1|ruled`), das neben jeder Quellseite eine liniierte Notizfläche freihält; ein `|notes`-Suffix, das dort stattdessen gesammelte `\Note`-Anmerkungen setzen würde, ist noch nicht implementiert. Ohne `layout` wird – wie bisher – jede Quellseite auf ein eigenes Zielblatt gesetzt.

`\tagpaxlastprefix`

Expandierbar. Liefert den intern vergebenen Präfix des zuletzt ausgeführten `\tagpaxinclude`-Aufrufs – z. B. um ihn extern zu protokollieren und in einem späteren Lauf wiederzuverwenden.

`\includepdf[⟨Optionen{⟨PDF-File⟩}⟩]`

Existiert nur bei gesetzter Paketoption `pdfpages`. Importiert direkt das angegeben File.

`pages = {⟨Seitenbereich⟩}`

Voreinstellung: -

Seitenauswahl, kein anderer Wert möglich.

`pagecommand = {⟨Ausdruck⟩}`

Voreinstellung: `{\thispagestyle{empty}}`

Deklariert L^AT_EX-Befehle, die für jede Seite ausgeführt werden.

Andere `PDFPAGES`-Optionen sind nicht gültig.

`\tagpaxsetup{⟨Optionen⟩}`

Legt die globalen Import- und Navigationsoptionen fest.

`heading-map = {⟨Zuordnung⟩}`

`toc-depth = {⟨Headerlevel⟩}`

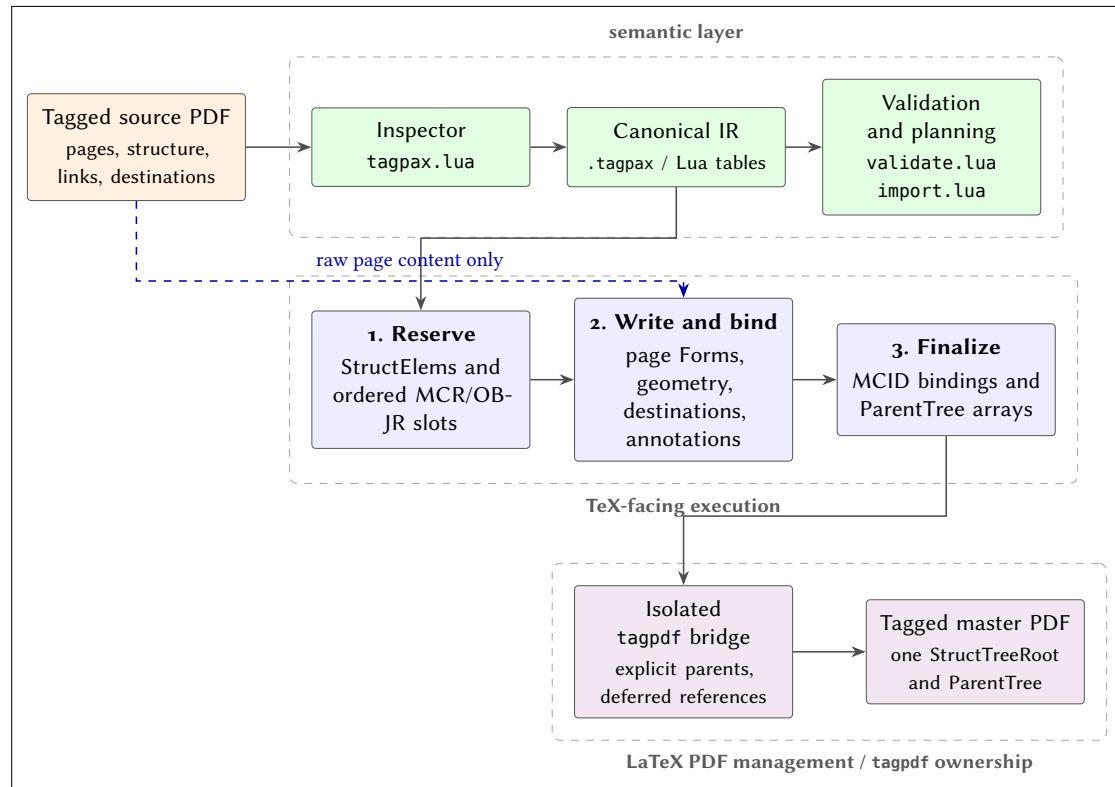
`bookmark-depth = {⟨Headerlevel⟩}` Siehe Abschnitt 2.3

3 Developer manual

3.1 Entwicklungsstand

Der aktuelle Stand ist für Experimente mit selbst erzeugten, korrekt getaggten Dokumenten geeignet, aber noch nicht sehr robust. Ein semantischer Roundtrip wird durch `l3build check -c roundtrip` geprüft. Seitenstream-MCRs, Link-Annotationen und lokale Ziele werden rekonstruiert. Explizite verschachtelte Form-XObjects bleiben außerhalb des unterstützten Importprofils.

3.2 Architektur



Source semantics are read once by the inspector and then passed through a canonical IR. The page writer reopens the PDF only for page content and geometry; it does not rediscover structure or navigation.

Layer	Input	Responsibility
Inspector	PDF object graph	Normalise structure, streams, destinations and annotations
Canonical IR	Plain Lua tables	Stable semantic document graph
Transformations	Canonical IR	Unwrap Document, create contribution Part, navigation
Import plan	Transformed IR + bindings	Resolve source streams to target Forms
Backend plan	Import plan	Order PDF-writing operations
Backend	Backend plan	Emit Forms, StructElems, MCRs, OBJRs and ParentTree entries

The architectural rules are:

- semantic processing *after* inspection use the canonical IR;
- PDF object numbers are not semantic identifiers;
- MCIDs are retained and remain local to their content stream;
- transformations are separated from PDF-writing operations;

- the central L^AT_EXtagging backend remains the sole owner of the ParentTree.

3.3 Lua API

The Lua modules form a supported developer interface. Functions documented in this section operate on plain Lua tables and do not require TeX unless stated otherwise. Callers should treat undocumented fields as private.

3.3.1 Inspector: *tagpax-inspect*

`inspect.from_pdf(filename, output?)`

Reads a tagged PDF with LuaTeX's `pdfc` library, normalises it and returns canonical IR. If output is supplied, the canonical serialisation is written as well.

`inspect.from_file(filename)`

Reads a serialised `.tagpax` file.

`inspect.summary(ir)`

Returns stable summary values for diagnostics and tests.

3.3.2 IR module: *tagpax-ir*

`ir.new()`

Creates an empty canonical IR value.

`ir.read(filename)`

Parses canonical line-oriented serialisation.

`ir.count_nodes(ir)`

Returns the number of structure nodes.

3.3.3 Validation: *tagpax-validate*

`validate.validate(ir)`

Checks referential integrity, stream bindings, root nodes, child order and heading references. On failure, errors is an ordered list.

3.3.4 Import planning: *tagpax-import*

`import.plan(ir, bindings, options?)`

Converts semantic IR into a backend-independent import plan. Page streams use `bindings.pages[page]`; explicit streams use `bindings.streams[id]`.

`import.assert_resolved(plan)`

Reports unresolved stream bindings. Explicit `/Stm` bindings are never redirected silently to a page Form.

3.3.5 LuaTeX importer: *tagpax-luatex*

`luatex.write_page(filename, page, structparents, stream_id, max_width?, max_height?)`
 Writes one source page as a fresh FormXObject, adds /StructParents, and reports its target object reference.

`luatex.page_geometry(filename, page, max_width?, max_height?)`
 Returns the shared crop, scale and rotation transformation used for the page Form, destination coordinates and annotation rectangles.

3.3.6 Native importer: *tagpax-native*

`native.emit_page_imports(pdf, irfile, prefix)`
 Emits one controlled page import operation per source page. This function is TeX-facing and requires LuaTeX. *<prefix>* namespaces the page-stream ids so multiple imports into one master document do not collide.

3.3.7 Backend prototype: *tagpax-backend*

`backend.emit_reservations(filename, prefix)`
 Reserves explicit-parent StructElems and source-ordered MCR/OBJR kid slots before page objects exist. *<prefix>* namespaces MCR stream ids so multiple imports into one master document do not collide.

`backend.emit_bindings(filename, prefix)`
 Registers the MCID bindings and commits external ParentTree arrays after pages and annotations have supplied their target object references. *<prefix>* must match the value passed to `backend.emit_reservations`.

3.4 Canonical intermediate representation

The IR is the normative semantic API. A .tagpax file is its canonical transport serialisation. IR version 1 contains source metadata, streams, structure nodes, ordered child relations, roots, headings, destinations and supported link annotations.

Record	Required fields	Meaning
source	file, pages	source-document identity
stream	id, kind	page or explicit object stream
node	id, role	structure element
kid	parent, index, kind	ordered child relation
heading	node, role, page, text	navigation data
destination	id, page, view	named target and view parameters
annotation	id, page, action, rect	link geometry and action data
root	index, node	root ordering

3.5 Backend

The backend first reserves all structure elements and source-ordered kid slots. It then writes Form XObjects and annotation overlays, binding their late PDF object references into the reserved slots. Finally it commits the MCR and annotation mappings to the central ParentTree. This three-phase order is necessary because LuaTeX allocates Form and annotation objects during page construction, but appending children only at that time would lose the source /K order. Imported annotations receive /StructParent and /OBJR entries owned by their original imported Link element; tagpdf's automatic replacement-Link mechanism is suppressed for these overlays. The bridge must not mutate the global `TAGPDF` structure stack.

3.6 Implementation

3.6.1 `tagpax.sty`

```

1 \NeedsTeXFormat{LaTeX2e}[2024-11-01]
2 \ProvidesExplPackage{tagpax}
3 {2026-10-22}
4 {v0.8.6}
5 {import of tagged PDF documents}
6 \RequirePackage{tagpax-tagpdf-bridge}
7 \ExplSyntaxOn
8 \bool_new:N \g__tagpax_pdfpages_compat_bool
9 \keys_define:nn {tagpax/package}
10 {
11   pdfpages .bool_gset:N = \g__tagpax_pdfpages_compat_bool,
12   pdfpages .default:n = true,
13 }
14 \ProcessKeyOptions[tagpax/package]
15 \msg_new:nnn {tagpax}{luatex-required}{tagpax~currently~requires~LuaLaTeX.}
16 \msg_new:nnn {tagpax}{metadata-required}{Activate~PDF~management/tagging~with~}
17   ↪ \string\DocumentMetadata{tagging=on}.}
18 \msg_new:nnn {tagpax}{backend-missing}{External-MCR~backend~not~installed;~}
19   ↪ navigation~is~generated,~structure~attachment~is~not~yet~written.}
20 \sys_if_engine_luatex:F {\msg_fatal:nn{tagpax}{luatex-required}}
21 \pdfmanagement_if_active:F {\msg_warning:nn{tagpax}{metadata-required}}
22 \seq_new:N \l__tagpax_headings_seq
23 \tl_new:N \l__tagpax_toc_depth_tl
24 \tl_new:N \l__tagpax_bookmark_depth_tl
25 \int_new:N \l__tagpax_toc_depth_int
26 \int_new:N \l__tagpax_bookmark_depth_int
27 \tl_new:N \l__tagpax_ir_tl
28 \tl_new:N \l__tagpax_file_tl
29 \tl_new:N \l__tagpax_pagecommand_tl
30 \tl_new:N \l__tagpax_layout_tl

```

```

29 \tl_new:N \l__tagpax_import_id_tl
30 \tl_new:N \l__tagpax_import_stream_key_tl
31 \int_new:N \g__tagpax_import_int
32 \tl_new:N \g__tagpax_last_prefix_tl
33 \prop_new:N \g__tagpax_heading_map_prop
34 \prop_gput_from_keyval:Nn \g__tagpax_heading_map_prop
    ↪ {H2=chapter,H3=section,H4=subsection,H5=subsubsection,H6=paragraph}
35 \keys_define:nn {tagpax/setup}{
36   toc-depth .code:n={
37     \tl_set:Nn\l__tagpax_toc_depth_tl{#1}
38     \int_set:Nn\l__tagpax_toc_depth_int{\tl_tail:n{#1}}
39   },
40   bookmark-depth .code:n={
41     \tl_set:Nn\l__tagpax_bookmark_depth_tl{#1}
42     \int_set:Nn\l__tagpax_bookmark_depth_int{\tl_tail:n{#1}}
43   },
44   heading-map .code:n=\prop_gput_from_keyval:Nn \g__tagpax_heading_map_prop
    ↪ {#1},
45 }
46 \keys_set:nn {tagpax/setup}{toc-depth=H4,bookmark-depth=H5}
47 \NewDocumentCommand\tagpaxsetup{m}{\keys_set:nn{tagpax/setup}{#1}}
48 \cs_new_protected:Npn \tagpax_heading_record:nnnn #1#2#3#4
    ↪ {\seq_put_right:Nn\l__tagpax_headings_seq{{#1}{#2}{#3}{#4}}}
49 \cs_new:Npn \__tagpax_heading_num:n #1 {\tl_tail:n{#1}}
50 \cs_new_protected:Npn \__tagpax_build_addtotoc:N #1 {
51   \tl_clear:N #1
52   \seq_map_inline:Nn \l__tagpax_headings_seq {
53     \__tagpax_build_addtotoc_aux:nnnnN ##1 #1
54   }
55 }
56 \cs_new_protected:Npn \__tagpax_build_addtotoc_aux:nnnnN #1#2#3#4#5 {
57   \int_compare:nNnT
58     {\__tagpax_heading_num:n{#1}}
59     <
60     {\l__tagpax_toc_depth_int+1} {
61     \prop_get:NnNT \g__tagpax_heading_map_prop {#1}\l_tmpa_tl {
62       \tl_if_empty:NF #5 {\tl_put_right:Nn #5 {},}}
63       \tl_put_right:Ne #5 {#2,\l_tmpa_tl,{\exp_not:n{#4}},tagpax.#3}
64     }
65   }
66 }
67 \keys_define:nn {tagpax/include}{
68   ir .tl_set:N=\l__tagpax_ir_tl,
69   toc-depth .code:n={

```

```

70   \tl_set:Nn\l__tagpax_toc_depth_tl{#1}
71   \int_set:Nn\l__tagpax_toc_depth_int{\tl_tail:n{#1}}
72 },
73 bookmark-depth .code:n={
74   \tl_set:Nn\l__tagpax_bookmark_depth_tl{#1}
75   \int_set:Nn\l__tagpax_bookmark_depth_int{\tl_tail:n{#1}}
76 },
77 pagecommand .tl_set:N=\l__tagpax_pagecommand_tl,
78 layout .tl_set:N=\l__tagpax_layout_tl,
79 }
80 \msg_new:nnn {tagpax}{deprecated-command}
81 {Command~\token_to_str:N #1~is~deprecated;~use~\token_to_str:N #2~instead.}
82 \NewDocumentCommand\tagpaxextract{0{m}}{
83   \directlua{
84     require('tagpax').extract_command(
85       "\luaescapestring{#2}", "\luaescapestring{#1}")
86   }
87 }
88 \NewDocumentCommand\TagPaxExtract{0{m}}{
89   \msg_warning:nnnn {tagpax}{deprecated-command}
90   {\TagPaxExtract}{\tagpaxextract}
91   \tagpaxextract[#1]{#2}
92 }
93 \NewDocumentCommand\tagpaxinclude{0{m}}{
94   \group_begin:
95   \int_gincr:N \g__tagpax_import_int
96   \tl_set:Ne \l__tagpax_import_id_tl {\int_use:N\g__tagpax_import_int}
97   \tl_gset:Ne \g__tagpax_last_prefix_tl {\l__tagpax_import_id_tl}
98   \tl_set:Nn \l__tagpax_file_tl {#2}
99   \tl_set:Nn \l__tagpax_ir_tl {#2}
100   \regex_replace_once:nnN { \.pdf\Z } {.tagpax} \l__tagpax_ir_tl
101   \tl_clear:N \l__tagpax_pagecommand_tl
102   \tl_clear:N \l__tagpax_layout_tl
103   \keys_set:nn {tagpax/include}{#1}
104   \directlua{require('tagpax-backend').emit_reservations(
105     [[\luaescapestring{\l__tagpax_ir_tl}]],
106     [[\luaescapestring{\l__tagpax_import_id_tl}]]))}
107   \directlua{require('tagpax-native').emit_page_imports(
108     [[\luaescapestring{#2}]], [[\luaescapestring{\l__tagpax_ir_tl}]],
109     [[\luaescapestring{\l__tagpax_import_id_tl}]],
110     [[\luaescapestring{\l__tagpax_layout_tl}]]))}
111   \directlua{require('tagpax-backend').emit_bindings(
112     [[\luaescapestring{\l__tagpax_ir_tl}]],
113     [[\luaescapestring{\l__tagpax_import_id_tl}]]))}

```

```

114 \group_end:
115 }
116 \cs_new:Npn \tagpaxlastprefix { \g__tagpax_last_prefix_tl }
117
118 \NewDocumentCommand\tagpaxroundtripinclude{0{}m}
119 {\tagpaxinclude[#1]{#2}}
120

```

Der Migrationsalias bleibt Teil von `tagpax` und damit lediglich eine zweite Oberfläche für denselben Seitenimport.

```

121 \msg_new:nnn {tagpax}{pdfpages-pages}
122 {The~tagpax~pdfpages~compatibility~option~supports~only~pages=-.}
123 \msg_new:nnn {tagpax}{pdfpages-option}
124 {The~pdfpages~option~'#1'~is~not~supported~by~the~native~tagpax~importer.}
125 \msg_new:nnn {tagpax}{pdfpages-conflict}
126 {
127   The~tagpax~pdfpages~compatibility~option~cannot~be~used~with~the~real~
128   pdfpages~package.~The~compatibility~\string\includepdf~is~disabled;~
129   use~\string\tagpaxinclude~for~semantic~imports.
130 }
131 \msg_new:nnn {tagpax}{includepdf-defined}
132 {
133   Command~\string\includepdf~already~exists.~The~tagpax~pdfpages~
134   compatibility~option~is~disabled;~use~\string\tagpaxinclude.
135 }
136 \tl_new:N \l__tagpax_pdfpages_pages_tl
137 \tl_new:N \l__tagpax_pdfpages_pagecommand_tl
138 \keys_define:nn {tagpax/pdfpages}
139 {
140   pages .tl_set:N = \l__tagpax_pdfpages_pages_tl,
141   pagecommand .tl_set:N = \l__tagpax_pdfpages_pagecommand_tl,
142   nup .code:n = \msg_error:nnn {tagpax}{pdfpages-option}{nup},
143   booklet .code:n = \msg_error:nnn {tagpax}{pdfpages-option}{booklet},
144   signature .code:n = \msg_error:nnn {tagpax}{pdfpages-option}{signature},
145 }
146
147 \cs_new_protected:Npn \__tagpax_includepdf:nn #1#2
148 {
149   \group_begin:
150     \tl_set:Nn \l__tagpax_pdfpages_pages_tl {-}
151     \tl_clear:N \l__tagpax_pdfpages_pagecommand_tl
152     \keys_set:nn {tagpax/pdfpages} {#1}
153     \str_if_eq:VnF \l__tagpax_pdfpages_pages_tl {-}
154       {\msg_error:nn {tagpax}{pdfpages-pages}}

```

```

155     \tagpaxinclude
156     [pagecommand=\l__tagpax_pdfpages_pagecommand_tl] {#2}
157   \group_end:
158 }
159
160 \cs_new_protected:Npn \__tagpax_disable_includepdf:
161 {
162   \bool_if:NT \g__tagpax_pdfpages_compat_bool
163   {
164     \msg_warning:nn {tagpax}{pdfpages-conflict}
165     \cs_undefine:N \includepdf
166     \bool_gset_false:N \g__tagpax_pdfpages_compat_bool
167   }
168 }
169
170 \bool_if:NT \g__tagpax_pdfpages_compat_bool
171 {
172   \IfPackageLoadedTF {pdfpages}
173   {
174     \msg_warning:nn {tagpax}{pdfpages-conflict}
175     \bool_gset_false:N \g__tagpax_pdfpages_compat_bool
176   }
177   {
178     \cs_if_exist:NTF \includepdf
179     {
180       \msg_warning:nn {tagpax}{includepdf-defined}
181       \bool_gset_false:N \g__tagpax_pdfpages_compat_bool
182     }
183     {
184       \NewDocumentCommand \includepdf { 0{ } m }
185       { \__tagpax_includepdf:nn {#1}{#2} }

```

Wird `pdfpages` später geladen, gibt `tagpax` den öffentlichen Befehl zuvor frei; die eindeutige native `\tagpaxinclude`-Schnittstelle bleibt.

```

186     \AddToHook {package/pdfpages/before}
187     { \__tagpax_disable_includepdf: }
188   }
189 }
190 }
191 \ExplSyntaxOff

```

3.6.2 tagpdf-bridge

```

1 \NeedsTeXFormat{LaTeX2e}[2024-11-01]

```

```

2 \ProvidesExplPackage{tagpax-tagpdf-bridge}
3 {2026-10-22}
4 {v0.8.6}
5 {Non-stack tagpdf bridge for tagpax}
6 \ExplSyntaxOn
7 \msg_new:nnn {tagpax}{bridge-incompatible}
8   {The~installed~tagpdf~does~not~provide~the~internals~required~by~the~tagpax~}
9   ↳ bridge.}
10 \msg_new:nnn {tagpax}{bridge-stream-unknown}
11   {External~stream~'#1'~has~not~been~allocated.}
12 \msg_new:nnn {tagpax}{bridge-stream-committed}
13   {External~stream~'#1'~has~already~been~committed.}
14 \cs_if_exist:NF \__tag_struct_prop_gput:nnn
15   { \msg_fatal:nn {tagpax}{bridge-incompatible} }
16 \prop_new:N \g__tagpax_bridge_stream_key_prop
17 \prop_new:N \g__tagpax_bridge_stream_committed_prop
18 \cs_generate_variant:Nn \prop_gput:Nnn { cee, Nee }
19 \seq_new:N \g__tagpax_bridge_stream_seq
20 \tl_new:N \l__tagpax_bridge_role_ns_tl
21 \tl_new:N \l__tagpax_bridge_role_tag_tl
22 \tl_new:N \l__tagpax_bridge_role_ns_resolved_tl
23 \int_new:N \l__tagpax_bridge_max_mcid_int
24 \tl_new:N \l__tagpax_bridge_parenttree_tl
25 \tl_new:N \l__tagpax_bridge_stream_key_tl
26 \cs_new_protected:Npn \tagpax_tag_struct_new:nnN #1#2#3
27   {
28     \int_gincr:N \c@g__tag_struct_abs_int
29     \tl_set:Nc #3 { \int_use:N \c@g__tag_struct_abs_int }
30     \__tag_prop_new:c { g__tag_struct_\tl_use:N #3 _prop }
31     \__tag_seq_new:c { g__tag_struct_kids_\tl_use:N #3 _seq }
32     \pdf_object_new_indexed:nn { __tag/struct } { \tl_use:N #3 }
33     \__tag_struct_prop_gput:nne { \tl_use:N #3 } { Type } { /StructElem }

```

Importierte Rollen werden zuerst in ihrem eigenen L^AT_EX-Namensraum aufgelöst, damit `tagpdf` sie auf native Rollen abbildet. Nur unbekannte Rollen verwenden den `pdf/pdf2`-Fallback.

```

34 \prop_get:NnNTF \g__tag_role_tags_NS_prop {#1} \l__tagpax_bridge_role_ns_tl
35   { }
36   {
37     \pdf_version_compare:NnTF < {2.0}
38       { \tl_set:Nn \l__tagpax_bridge_role_ns_tl {pdf} }
39       { \tl_set:Nn \l__tagpax_bridge_role_ns_tl {pdf2} }
40   }
41 \__tag_role_get:nnNN

```

```

42     {#1}
43     {\__tagpax_bridge_role_ns_tl}
44     \__tagpax_bridge_role_tag_tl
45     \__tagpax_bridge_role_ns_resolved_tl
46     \tl_if_empty:NT \__tagpax_bridge_role_tag_tl
47     { \tl_set:Nn \__tagpax_bridge_role_tag_tl {#1} }
48     \tl_if_empty:NT \__tagpax_bridge_role_ns_resolved_tl
49     { \tl_set_eq:NN \__tagpax_bridge_role_ns_resolved_tl
      ↪ \__tagpax_bridge_role_ns_tl }
50     \__tag_struct_set_tag_info:eoo
51     { \tl_use:N #3 }
52     { \__tagpax_bridge_role_tag_tl }
53     { \__tagpax_bridge_role_ns_resolved_tl }
54     \__tag_struct_prop_gput:nne
55     { \tl_use:N #3 }
56     { rolemap }
57     { {\__tagpax_bridge_role_tag_tl}{\__tagpax_bridge_role_ns_resolved_tl} }
58     \__tag_struct_prop_gput:nne
59     { \tl_use:N #3 }
60     { parentrole }
61     { {\__tagpax_bridge_role_tag_tl}{\__tagpax_bridge_role_ns_resolved_tl} }
62     \__tag_struct_prop_gput:nne
63     { \tl_use:N #3 }
64     { parentnum }
65     {#2}
66     \__tag_struct_kid_struct_gput_right:ee {#2} {\tl_use:N #3}
67   }
68
69   \cs_new_protected:Npn \tagpax_tag_struct_append_mcr:nnn #1#2#3
70   {
71     \__tag_seq_gput_right:ce
72     { g__tag_struct_kids_#1_seq }
73     {
74       <<
75       /Type~/MCR
76       /Stm~#2
77       /MCID~\int_eval:n{#3}
78       >>
79     }
80   }
81
82   \cs_new_protected:Npn \tagpax_tag_stream_new:nN #1#2
83   {
84     \prop_if_in:NnTF \g__tagpax_bridge_stream_key_prop {#1}

```

```

85     { \prop_get:NnN \g__tagpax_bridge_stream_key_prop {#1} #2 }
86     {
87       \int_gincr:N \c@g__tag_parenttree_obj_int
88       \tl_set:Ne #2 { \int_use:N \c@g__tag_parenttree_obj_int }
89       \prop_gput:NnV \g__tagpax_bridge_stream_key_prop {#1} #2
90       \prop_new:c { g__tagpax_bridge_stream_\tl_use:N #2 _prop }
91       \seq_gput_right:Nn \g__tagpax_bridge_stream_seq {#1}
92     }
93   }
94
95   \cs_new:Npn \tagpax_tag_stream_structparents:n #1
96     { \prop_item:Nn \g__tagpax_bridge_stream_key_prop {#1} }
97
98   \cs_new:Npn \tagpax_tag_stream_attr:n #1
99     { /StructParents~\tagpax_tag_stream_structparents:n {#1} }
100
101   \cs_new_protected:Npn \tagpax_tag_stream_register:nnn #1#2#3
102     {
103       \prop_get:NnNTF \g__tagpax_bridge_stream_key_prop {#1}
104       ↪ \l__tagpax_bridge_stream_key_tl
105       {
106         \prop_gput:cee
107         { g__tagpax_bridge_stream_\l__tagpax_bridge_stream_key_tl _prop }
108         { \int_eval:n {#2} }
109         {#3}
110       }
111       { \msg_error:nnn {tagpax}{bridge-stream-unknown}{#1} }
112     }
113
114   \cs_new_protected:Npn \tagpax_tag_stream_commit:n #1
115     {
116       \prop_if_in:NnTF \g__tagpax_bridge_stream_committed_prop {#1}
117       { \msg_warning:nnn {tagpax}{bridge-stream-committed}{#1} }
118       {
119         \prop_get:NnNTF \g__tagpax_bridge_stream_key_prop {#1} \l_tmpa_tl
120         {
121           \int_set:Nn \l__tagpax_bridge_max_mcid_int {-1}
122           \prop_map_inline:cn { g__tagpax_bridge_stream_\l_tmpa_tl _prop }
123           {
124             \int_set:Nn \l__tagpax_bridge_max_mcid_int
125             { \int_max:nn {\l__tagpax_bridge_max_mcid_int}{##1} }
126           }
127           \tl_clear:N \l__tagpax_bridge_parenttree_tl
128           \int_step_inline:nnnn {0}{1}{\l__tagpax_bridge_max_mcid_int}

```

```

128         {
129             \prop_get:cnNTF
130             { g__tagpax_bridge_stream_\l_tmpa_tl _prop }
131             {##1}
132             \l_tmpb_tl
133             {
134                 \tl_put_right:Ne \l__tagpax_bridge_parenttree_tl
135                 { \pdf_object_ref_indexed:nn {__tag/struct}{\l_tmpb_tl}~ }
136             }
137             { \tl_put_right:Nn \l__tagpax_bridge_parenttree_tl { null~ } }
138         }
139         \tl_gput_right:Ne \g__tag_parenttree_objr_tl
140         {
141             \l_tmpa_tl~[~\l__tagpax_bridge_parenttree_tl]^J
142         }
143         \prop_gput:Nnn \g__tagpax_bridge_stream_committed_prop {#1} {true}
144     }
145     { \msg_error:nnn {tagpax}{bridge-stream-unknown}{#1} }
146 }
147 }
148
149
150 \prop_new:N \g__tagpax_backend_node_prop
151 \prop_new:N \g__tagpax_backend_form_prop
152 \prop_new:N \g__tagpax_backend_page_stream_prop
153 \prop_new:N \g__tagpax_backend_objr_prop
154 \tl_new:N \g__tagpax_backend_wrapper_tl
155 \tl_new:N \l__tagpax_backend_annotation_tl
156 \tl_new:N \l__tagpax_backend_annotation_parent_tl
157 \tl_new:N \l__tagpax_backend_annotation_action_tl
158 \tl_new:N \l__tagpax_backend_annotation_key_tl
159
160 \cs_new_protected:Npn \TagPaxBackendForm #1#2#3
161 {
162     \prop_gput:Nnn \g__tagpax_backend_form_prop {#1} {#2}
163     \prop_gput:Nnn \g__tagpax_backend_page_stream_prop {#1} {#3}
164 }
165
166 \cs_new_protected:Npn \TagPaxPageDestination #1#2
167 { \pdf_destination:nn {#1}{#2} }
168
169 \cs_new_protected:Npn \TagPaxPointDestination #1#2#3#4#5
170 {
171     \kern-\dim_eval:n{#1sp}

```

```

172 \hbox_to_wd:nn {#1sp}
173 {
174   \kern#2sp
175   \raise#3sp\hbox{\pdf_destination:nn {#4}{#5}}
176   \hss
177 }
178 }
179
180 \cs_new_protected:Npn \TagPaxRectangleDestination #1#2#3#4#5#6
181 {
182   \kern-\dim_eval:n{#1sp}
183   \hbox_to_wd:nn {#1sp}
184   {
185     \kern#2sp
186     \raise#3sp\hbox
187       {\pdf_destination:nnnn {#6}{#4sp}{#5sp}{0pt}}
188     \hss
189   }
190 }
191
192 \cs_new_protected:Npn \TagPaxNavigationHeading #1#2#3
193 {
194   \int_compare:nNnT
195     {\__tagpax_heading_num:n{#1}}
196     <
197     {\l__tagpax_toc_depth_int+1}
198   {
199     \prop_get:NnNT \g__tagpax_heading_map_prop {#1}\l_tmpa_tl
200     {
201       \group_begin:
202         \cs_if_exist:NT \@currentHref {\tl_set:Nn\@currentHref{#3}}
203         \addcontentsline{toc}{\l_tmpa_tl}{#2}
204       \group_end:
205     }
206   }
207   \int_compare:nNnT
208     {\__tagpax_heading_num:n{#1}}
209     <
210     {\l__tagpax_bookmark_depth_int+1}
211   {
212     \cs_if_exist:NT \bookmark
213     {
214       \bookmark[
215         level=\int_eval:n{\__tagpax_heading_num:n{#1}-2},

```

```

216         dest={#3}
217     ]{#2}
218 }
219 }
220 }
221
222 \NewTaggingSocketPlug{link/before}{tagpax-import}
223 {
224     \pdfannot_dict_put:nne
225     {link/\l__tagpax_backend_annotation_action_tl}
226     {StructParent}
227     {\l__tagpax_backend_annotation_key_tl}
228 }
229 \NewTaggingSocketPlug{link/after}{tagpax-import}
230 {
231     \__tag_property_record:en
232     {@tagpax@objr@page@\l__tagpax_backend_annotation_key_tl}
233     {tagabspage}
234     \pdf_object_unnamed_write:ne {dict}
235     {
236         /Type/OBJR
237         /Obj~\pdfannot_link_ref_last:
238         /Pg~\pdf_pageobject_ref:n
239         {
240             \property_ref:enn
241             {@tagpax@objr@page@\l__tagpax_backend_annotation_key_tl}
242             {tagabspage}{1}
243         }
244     }
245     \prop_gput:Nee \g__tagpax_backend_objr_prop
246     {\l__tagpax_backend_annotation_tl}
247     {\pdf_object_ref_last:}
248     \__tag_parenttree_add_objr:nn
249     {\l__tagpax_backend_annotation_key_tl}
250     {
251         \pdf_object_ref_indexed:nn
252         {__tag/struct}{\l__tagpax_backend_annotation_parent_tl}
253     }
254     \int_gincr:N \c@g__tag_parenttree_obj_int
255 }
256
257 \cs_new_protected:Npn \__tagpax_annotation_begin:nn #1#2
258 {
259     \tl_set:Nn \l__tagpax_backend_annotation_tl {#1}

```

```

260 \tl_set:Nn \l__tagpax_backend_annotation_action_tl {#2}
261 \prop_get:NnN \g__tagpax_backend_annotation_parent_prop
262 {#1}\l__tagpax_backend_annotation_parent_tl
263 \tl_set:Ne \l__tagpax_backend_annotation_key_tl
264 {\int_use:N \c@g__tag_parenttree_obj_int}

```

Importierte Links besitzen bereits einen expliziten Quell-Elternknoten und werden unten gebunden. Der Lua-Callback ignoriert sie, damit kein zweites OBJR im Link-Bin entsteht.

```

265 \lua_now:n
266 {tex.setattribute(luatexbase.attributes.g__tag_structnum_attr,-1)}
267 \AssignTaggingSocketPlug{link/before}{tagpax-import}
268 \AssignTaggingSocketPlug{link/after}{tagpax-import}
269 }
270
271 \cs_new_protected:Npn \TagPaxGotoOverlay #1#2#3#4#5#6#7
272 {
273 \group_begin:
274 \__tagpax_annotation_begin:nn {#6}{GoTo}
275 \kern-\dim_eval:n{#1sp}
276 \hbox_to_wd:nn {#1sp}
277 {
278 \kern#2sp
279 \raise#3sp\hbox
280 {
281 \pdfannot_link_goto_begin:nw {#7}
282 \hbox_to_wd:nn {#4sp}{\rule{0pt}{#5sp}\hfill}
283 \pdfannot_link_goto_end:
284 }
285 \hss
286 }
287 \group_end:
288 }
289
290 \cs_new_protected:Npn \__tagpax_action_overlay:nnnnnnnn #1#2#3#4#5#6#7#8
291 {
292 \group_begin:
293 \__tagpax_annotation_begin:nn {#6}{#7}
294 \kern-\dim_eval:n{#1sp}
295 \hbox_to_wd:nn {#1sp}
296 {
297 \kern#2sp
298 \raise#3sp\hbox
299 {
300 \pdfannot_link_begin:nnw {#7}{#8}

```

```

301         \hbox_to_wd:nn {#4sp}{\rule{0pt}{#5sp}\hfill}
302         \pdfannot_link_end:n {#7}
303     }
304     \hss
305 }
306 \group_end:
307 }
308
309 \cs_new_protected:Npn \TagPaxURIOverlay #1#2#3#4#5#6#7
310 {
311     \__tagpax_action_overlay:nnnnnnnn
312     {#1}{#2}{#3}{#4}{#5}{#6}{URI}
313     {/A~<</Type~/Action~/S~/URI~/URI~<#7>>>}
314 }
315
316 \cs_new_protected:Npn \TagPaxGoToRNameOverlay #1#2#3#4#5#6#7#8
317 {
318     \__tagpax_action_overlay:nnnnnnnn
319     {#1}{#2}{#3}{#4}{#5}{#6}{GoToR}
320     {/A~<</Type~/Action~/S~/GoToR~/F~<#7>~/D~<#8>>>}
321 }
322
323 \cs_new_protected:Npn \TagPaxGoToRPageOverlay #1#2#3#4#5#6#7#8#9
324 {
325     \__tagpax_action_overlay:nnnnnnnn
326     {#1}{#2}{#3}{#4}{#5}{#6}{GoToR}
327     {/A~<</Type~/Action~/S~/GoToR~/F~<#7>~/D~[#8~/#9]>>}
328 }
329
330 \cs_new_protected:Npn \TagPaxImportOnePage #1#2#3#4#5
331 {
332     \tagpax_tag_stream_new:nN {#3} \l__tagpax_import_stream_key_tl
333     \clearpage
334     \thispagestyle{empty}
335     \tl_use:N \l__tagpax_pagecommand_tl
336     \directlua{require('tagpax-native').emit_page_navigation(
337         [[\luaescapestring{#4}]],#2,[[\luaescapestring{#5}]])}
338     \noindent\hbox to \linewidth
339     {\hss\directlua{require('tagpax-luatex').write_page(
340         [[\luaescapestring{
341             ↪ #1}]],#2,\l__tagpax_import_stream_key_tl,[[\luaescapestring{#3}]],
342         [[\luaescapestring{#4}]],[[\luaescapestring{#5}]]],
343         \number\textwidth,\number\textheight)}\hss}
344     \clearpage

```

```

344 }
345

```

Rasterimposition (@1) hält denselben Vertrag pro Quellseite wie @2: ein frischer @3-Kontext, ein eigener @4-Aufruf. Nur die Platzierung ändert sich – von eine volle Seite zu eine Rasterzelle fester Größe. Zellgröße und Zellzahl pro Blatt sind Paketstate (@5/@6, @7/@8), nicht Makroparameter, weil mehrere unabhängige T_EX-Aufrufe pro physischem Blatt zusammenwirken müssen, ohne dass jeder Aufruf die volle Geometrie erneut mitführt. `\TagPaxGridPageBegin\TagPaxImportOnePage`

```

StructParentswrite_page\l__tagpax_grid_rows_int\l__tagpax_grid_cols_int\l__tagpax_grid_cellwidth_dim
346 \int_new:N \l__tagpax_grid_rows_int
347 \int_new:N \l__tagpax_grid_cols_int
348 \int_new:N \g__tagpax_grid_col_int
349 \dim_new:N \l__tagpax_grid_cellwidth_dim
350 \dim_new:N \l__tagpax_grid_cellheight_dim
351 \dim_new:N \l__tagpax_grid_imagewidth_dim
352 \dim_new:N \l__tagpax_grid_notewidth_dim
353 \box_new:N \g__tagpax_grid_page_box
354 \box_new:N \g__tagpax_grid_row_box
355 \box_new:N \l__tagpax_grid_cell_box
356

```

Ohne Zwischenraum stoßen benachbarte Zellen direkt aneinander; einzelne Quellseiten sind dann kaum als solche erkennbar. Der Wert ist bewusst klein und fest, kein Schlüssel: er soll Zellen nur sichtbar trennen, nicht Platz für Inhalt schaffen.

```

357 \dim_new:N \l__tagpax_grid_gutter_dim
358 \dim_set:Nn \l__tagpax_grid_gutter_dim {3mm}
359

```

Zeilenhöhen summieren sich exakt zu `\textheight`. Ohne Abzug bliebe kein Platz für den von T_EX oberhalb des ersten Elements jeder Seite obligatorisch eingefügten `\topskip`-Klebstoff – die Seite überliefe dann um genau diesen Betrag, und der Seitenumbruchalgorithmus verschöbe das gesamte, unteilbare Rasterblatt auf die nächste Seite (eine scheinbar leere Seite wäre die Folge).

```

360 \cs_new_protected:Npn \TagPaxGridPageBegin #1#2
361 {
362   \clearpage
363   \thispagestyle{empty}
364   \tl_use:N \l__tagpax_pagecommand_tl
365   \int_set:Nn \l__tagpax_grid_rows_int {#1}
366   \int_set:Nn \l__tagpax_grid_cols_int {#2}
367   \dim_set:Nn \l__tagpax_grid_cellwidth_dim
368     {
369     ( \textwidth

```

```

370         - \l__tagpax_grid_gutter_dim * ( \l__tagpax_grid_cols_int - 1 ) )
371       / \l__tagpax_grid_cols_int
372     }
373   \dim_set:Nn \l__tagpax_grid_cellheight_dim
374     {
375     ( \textheight - \topskip
376       - \l__tagpax_grid_gutter_dim * ( \l__tagpax_grid_rows_int - 1 ) )
377     / \l__tagpax_grid_rows_int
378     }
379   \box_gclear:N \g__tagpax_grid_page_box
380   \box_gclear:N \g__tagpax_grid_row_box
381   \int_gzero:N \g__tagpax_grid_col_int
382 }
383

```

Wird auch am Blattende unbedingt aufgerufen: eine unvollständige letzte Zeile (Seitenzahl kein Vielfaches der Rastergröße) bleibt dann einfach mit leeren Zellen rechts/unten, statt verworfen zu werden.

```

384 \cs_new_protected:Npn \__tagpax_grid_flush_row:
385 {
386   \int_compare:nNnT {\g__tagpax_grid_col_int} > {0}
387   {
388     \box_if_empty:NTF \g__tagpax_grid_page_box
389     {
390       \vbox_gset:Nn \g__tagpax_grid_page_box
391       { \box_use:N \g__tagpax_grid_row_box }
392     }
393     {
394       \vbox_gset:Nn \g__tagpax_grid_page_box
395       {
396         \box_use:N \g__tagpax_grid_page_box
397         \kern \l__tagpax_grid_gutter_dim
398         \box_use:N \g__tagpax_grid_row_box
399       }
400     }
401     \box_gclear:N \g__tagpax_grid_row_box
402     \int_gzero:N \g__tagpax_grid_col_int
403   }
404 }
405
406 \cs_new_protected:Npn \TagPaxGridPageEnd
407 {
408   \__tagpax_grid_flush_row:
409   \noindent \box_use:N \g__tagpax_grid_page_box

```

```

410     \clearpage
411 }
412
413 \cs_new_protected:Npn \__tagpax_grid_append_cell:
414 {
415     \box_if_empty:NTF \g__tagpax_grid_row_box
416     {
417         \hbox_gset:Nn \g__tagpax_grid_row_box
418         { \box_use:N \l__tagpax_grid_cell_box }
419     }
420     {
421         \hbox_gset:Nn \g__tagpax_grid_row_box
422         {
423             \box_use:N \g__tagpax_grid_row_box
424             \kern \l__tagpax_grid_gutter_dim
425             \box_use:N \l__tagpax_grid_cell_box
426         }
427     }
428     \int_gincr:N \g__tagpax_grid_col_int
429     \int_compare:nNnT {\g__tagpax_grid_col_int} = {\l__tagpax_grid_cols_int}
430     { \__tagpax_grid_flush_row: }
431 }
432

```

Zentriert vertikal in fester Zellhöhe (`\l__tagpax_grid_cellheight_dim`), unabhängig vom tatsächlichen, durch `write_page` bereits proportional herunterskalierten Seitenverhältnis der Quellseite – sonst würde jede Zeile an der Bildhöhe der jeweils enthaltenen Quellseiten driften.

```

433 \cs_new_protected:Npn \__tagpax_grid_image:nnnnnn #1#2#3#4#5#6
434 {
435     \tagpax_tag_stream_new:nN {#3} \l__tagpax_import_stream_key_tl
436     \directlua{require('tagpax-native').emit_page_navigation(
437         [[\luaescapestring{#4}]],#2,[[\luaescapestring{#5}]]})}
438     \vbox_to_ht:nn {\l__tagpax_grid_cellheight_dim}
439     {
440         \vfil
441         \hbox_to_wd:nn {#6}
442         {\hss\directlua{require('tagpax-luatex').write_page(
443             [[\luaescapestring{
444                 ↪ #1}]],#2,\l__tagpax_import_stream_key_tl,[[\luaescapestring{
445                 ↪ #3}]],,
446             [[\luaescapestring{#4}]],[[\luaescapestring{#5}]],
447             \number#6,\number\l__tagpax_grid_cellheight_dim)}\hss}
448         \vfil
449     }

```

```

448 }
449
450 \cs_new_protected:Npn \TagPaxImportGridPage #1#2#3#4#5
451 {
452   \dim_set_eq:NN \l__tagpax_grid_imagewidth_dim \l__tagpax_grid_cellwidth_dim
453   \vbox_set:Nn \l__tagpax_grid_cell_box
454     { \__tagpax_grid_image:nnnnnn
455       ↪ {#1}{#2}{#3}{#4}{#5}{\l__tagpax_grid_imagewidth_dim} }
456   \__tagpax_grid_append_cell:
457 }

```

Die Notizfläche zeichnet nur linierten Leerraum – kein Seiteninhalt wird dafür ausgelesen, siehe die entsprechende Design-Notiz zu zurückgestelltem Notizinhalt.

```

458 \dim_new:N \l__tagpax_grid_ruled_step_dim
459 \dim_set:Nn \l__tagpax_grid_ruled_step_dim {8mm}
460 \cs_new_protected:Npn \__tagpax_grid_ruled_lines:n #1
461 {
462   \hbox_to_wd:nn {#1}
463   {
464     \hfil
465     \vbox_to_ht:nn {\l__tagpax_grid_cellheight_dim}
466     {
467       \vfil
468       \prg_replicate:nn {5}
469         { \rule{0.8\l__tagpax_grid_notewidth_dim}{0.4pt} \vfil }
470     }
471     \hfil
472   }
473 }
474
475 \cs_new_protected:Npn \TagPaxImportGridPageRuled #1#2#3#4#5
476 {
477   \dim_set:Nn \l__tagpax_grid_imagewidth_dim
478     { 0.72 \l__tagpax_grid_cellwidth_dim }
479   \dim_set:Nn \l__tagpax_grid_notewidth_dim
480     { \l__tagpax_grid_cellwidth_dim - \l__tagpax_grid_imagewidth_dim }
481   \hbox_set:Nn \l__tagpax_grid_cell_box
482     {
483       \__tagpax_grid_image:nnnnnn
484       ↪ {#1}{#2}{#3}{#4}{#5}{\l__tagpax_grid_imagewidth_dim}
485       \__tagpax_grid_ruled_lines:n {\l__tagpax_grid_notewidth_dim}
486     }
487   \__tagpax_grid_append_cell:

```

```

487 }
488
489 \cs_new_protected:Npn \TagPaxBackendDocumentBegin
490 {
491   \seq_gclear:N \g__tagpax_bridge_stream_seq
492   \prop_gclear:N \g__tagpax_backend_node_prop
493   \prop_gclear:N \g__tagpax_backend_objr_prop
494   \prop_gclear:N \g__tagpax_backend_annotation_parent_prop
495   \tl_set_eq:NN \l_tmpa_tl \g__tag_struct_stack_current_tl
496   \tagpax_tag_struct_new:nnN {Part}{\l_tmpa_tl}\g__tagpax_backend_wrapper_tl
497   \prop_gput:NnV \g__tagpax_backend_node_prop
498   ↪ { @wrapper } \g__tagpax_backend_wrapper_tl
499 }
500 \cs_new_protected:Npn \TagPaxBackendDocumentEnd
501 {
502   \seq_map_inline:Nn \g__tagpax_bridge_stream_seq
503   { \tagpax_tag_stream_commit:n {##1} }
504 }
505 \cs_new_protected:Npn \TagPaxBackendNode #1#2#3
506 {
507   \prop_get:NnN \g__tagpax_backend_node_prop {#3}\l_tmpa_tl
508   \tagpax_tag_struct_new:nnN {#2}{\l_tmpa_tl}\l_tmpb_tl
509   \prop_gput:NnV \g__tagpax_backend_node_prop {#1}\l_tmpb_tl
510 }

```

Die Rolle wird auf sich selbst registriert statt via `\tagpdfsetup{role/new-tag=...}` auf `role` aufgelöst: `\tagpdfsetup` würde den importierten Rollennamen durch `role` ersetzen und damit gerade die Unterscheidung verwerfen, die die Quellstruktur getroffen hat. Die Selbstregistrierung genügt, um die unknown tag Warnung abzustellen; der separate Eintrag in `RoleMap` sorgt dafür, dass tag dennoch mit korrektem Mapping auf einen Standardtyp im finalen PDF erscheint. Nur bisher unbekannte Rollennamen werden registriert; eine bereits bekannte Rolle – ob nativ oder aus einer früheren `\tagpaxinclude` – bleibt unverändert, damit wiederholte Importe idempotent bleiben.

```

510 \cs_new_protected:Npn \TagPaxBackendRoleMap #1#2
511 {
512   \prop_get:NnNF \g__tag_role_tags_NS_prop {#1} \l_tmpa_tl
513   {
514     \prop_gput:Nnn \g__tag_role_tags_NS_prop {#1}{user}
515     \prop_gput:Nnn \g__tag_role_NS_user_prop {#1}{#1}{user}
516     \prop_gput:Nnn \g__tag_role_rolemap_prop {#1}{#2}
517   }
518 }
519 \prop_new:N \g__tagpax_backend_annotation_parent_prop
520 \cs_new:Npn \__tagpax_backend_form_ref:n #1

```

```

521 { \prop_item:Nn \g__tagpax_backend_form_prop {#1} }
522 \cs_new:Npn \__tagpax_backend_objr_ref:n #1
523 { \prop_item:Nn \g__tagpax_backend_objr_prop {#1} }
524 \cs_new_protected:Npn \TagPaxBackendReserveMCR #1#2#3#4#5
525 {
526   \prop_get:NnN \g__tagpax_backend_node_prop {#5}\l_tmpa_tl
527   \__tag_seq_gput_right:cn {g__tag_struct_kids_\l_tmpa_tl _seq}
528   {
529     <</Type~/MCR/MCID~\int_eval:n{#3}
530     /Stm~\__tagpax_backend_form_ref:n{#2}>>
531   }
532 }
533 \cs_new_protected:Npn \TagPaxBackendBindMCR #1#2#3#4#5
534 {
535   \prop_get:NnN \g__tagpax_backend_node_prop {#5}\l_tmpa_tl
536   \exp_args:NnnV \tagpax_tag_stream_register:nnn {#4}{#3}\l_tmpa_tl
537 }
538 \cs_new_protected:Npn \TagPaxBackendReserveOBJR #1#2
539 {
540   \prop_get:NnN \g__tagpax_backend_node_prop {#2}\l_tmpa_tl
541   \prop_gput:NnV \g__tagpax_backend_annotation_parent_prop {#1}\l_tmpa_tl
542   \__tag_seq_gput_right:cn {g__tag_struct_kids_\l_tmpa_tl _seq}
543   {\__tagpax_backend_objr_ref:n{#1}}
544 }
545 \cs_new_protected:Npn \TagPaxBackendUnsupportedStream #1#2
546 { \msg_error:nnn {tagpax}{bridge-stream-unknown}{#1} }
547

```

3.6.3 tagpax.lua

Version und Datum für die Kompatibilitätsprüfung.

```

1 local function package_info(filename)
2   local path = kpse and kpse.find_file(filename, "lua") or filename
3   local file, err = io.open(path or filename, "r")
4   if not file then
5     error("cannot read package metadata from " .. filename .. ": " ..
6       ↳ tostring(err))
7   end
8   local content, read_error = file:read("*all")
9   file:close()
10  if not content then
11    error("cannot read package metadata from " .. filename .. ": " ..
12      ↳ tostring(read_error))
13  end

```

```

12  local header = content:match("^%s*%-%%[%[(.-)%]%)")
13  if not header then
14      error("cannot find package metadata in " .. filename)
15  end
16  return {
17      version = header:match("\n%s*Version:%s*\n%s*([^\r\n]+)"),
18      date = header:match("\n%s*Date:%s*\n%s*([^\r\n]+)")
19  }
20  end
21  local M = package_info("tagpax.lua")
22

```

Diese beiden Module hängen selbst nicht von `tagpax.lua` ab; ein zirkulärer Require entsteht also nicht, und eager Laden am Dateianfang ist unbedenklich.

```

23  local ir_reader = require("tagpax-ir")
24  local validator = require("tagpax-validate")
25
26  local pdfc = assert(pdfc, "tagpax requires LuaTeX's pdfc library")
27

```

Transportkodierung Das zeilenorientierte Format bleibt diffbar; Prozentkodierung schützt Datensatzgrenzen vor Tabs, Zeilenumbrüchen und PDF-Strings.

```

28  local function pct(s)
29      s = tostring(s or "")
30      return (s:gsub("[^%w%-%.~]", function(c)
31          return string.format("%%%02X", string.byte(c))
32      end))
33  end
34
35  local function field(t, k)
36      return t[k] ~= nil and (k .. "=" .. pct(t[k])) or nil
37  end
38
39  local function write_record(f, kind, t)
40      local fields = { kind }
41      for _, key in ipairs(t.__order or {}) do
42          local encoded = field(t, key)
43          if encoded then fields[#fields + 1] = encoded end
44      end
45      f:write(table.concat(fields, "\t"), "\n")
46  end
47

```

Normalisierung der @1-Schnittstelle pdfc Diese Adapter kapseln die Typ- und Referenzkonventionen von LuaTeX; die Extraktion arbeitet dadurch mit semantischen Werten.

```

48 local function value(dict, key)
49     local typ, val, detail = pdf.getfromdictionary(dict, key)
50     return typ, val, detail
51 end
52
53 local function decode_pdf_string(s, hexadecimal)

```

Hex- und Literalstrings sind Bytefolgen. UTF-16BE wird per BOM erkannt; andere Kodierungen bleiben verlustfrei erhalten.

```

54     if hexadecimal then
55         s = tostring(s or ""):gsub("%s+", "")
56         s = s:gsub("(%x%x)", function(pair)
57             return string.char(tonumber(pair, 16))
58         end)
59     end
60     if s and #s >= 2 and s:byte(1) == 0xFE and s:byte(2) == 0xFF then
61         local chars = {}
62         local index = 3
63         while index + 1 <= #s do
64             local first = s:byte(index) * 256 + s:byte(index + 1)
65             index = index + 2
66             if first >= 0xD800 and first <= 0xDBFF and index + 1 <= #s then
67                 local second = s:byte(index) * 256 + s:byte(index + 1)
68                 if second >= 0xDC00 and second <= 0xDFFF then
69                     first = 0x10000 + (first - 0xD800) * 0x400 + second - 0xDC00
70                     index = index + 2
71                 end
72             end
73             chars[#chars + 1] = utf8.char(first)
74         end
75         return table.concat(chars)
76     end
77     return s
78 end
79
80 local function pdf_string(dict, key)
81     local typ, val, detail = value(dict, key)
82     if typ == 6 or typ == "string" then return decode_pdf_string(val, detail) end
83     return nil
84 end
85
86 local function pdf_name(dict, key)
87     local typ, val = value(dict, key)
88     if typ == 5 or typ == "name" then return val end

```

```

89     return nil
90 end
91
92 local function pdf_filespec(dict, key)
    /UF enthält den Unicode-Namen, /F den Kompatibilitätswert.
93     local _, item = value(dict, key)
94     local current = item
95     if type(current) == "string" then return current end
96     if pdf.type(current) == "pdf.reference" then
97         local _, resolved = pdf.getfromreference(current)
98         current = resolved
99     end
100    if pdf.type(current) == "pdf.dictionary" then
101        return pdf_string(current, "UF") or pdf_string(current, "F")
102    end
103    return nil
104 end
105
106 local function as_dict(v)

```

Struktureinträge können direkte Objekte oder indirekte Referenzen sein.

```

107     if pdf.type(v) == "pdf.dictionary" then return v end
108     if pdf.type(v) == "pdf.reference" then
109         local _, resolved = pdf.getfromreference(v)
110         if pdf.type(resolved) == "pdf.dictionary" then return resolved end
111     end
112 end
113
114 local function as_array(v)
115     if pdf.type(v) == "pdf.array" then return v end
116     if pdf.type(v) == "pdf.reference" then
117         local _, resolved = pdf.getfromreference(v)
118         if pdf.type(resolved) == "pdf.array" then return resolved end
119     end
120 end
121
122 local function ref_number(v)
123     if pdf.type(v) == "pdf.reference" then

```

`pdf.getfromreference` löst das Objekt auf, liefert dessen Nummer aber nicht zuverlässig; die Textdarstellung der Referenz enthält sie.

```

124     return tonumber(tostring(v):match("pdf%.reference%s+(%d+)"))
125 end

```

```

126     return nil
127 end

```

```

128
129 local function page_map(doc)

```

IR-Datensätze verwenden stabile einsbasierte Seitennummern, keine PDF-Objektnummern.

```

130     local pages = pdfc.pagestotable(doc)
131     local map = {}
132     for number, page in ipairs(pages) do map[page[3]] = number end
133     return map, #pages
134 end

```

```

135
136 local function array_values(array)
137     local result = {}
138     if not array then return result end
139     for index = 1, #array do
140         local _, item = pdfc.getfromarray(array, index)
141         result[#result + 1] = item
142     end
143     return result
144 end

```

```

145
146 local function name_tree(dict, target)

```

Der balancierte PDF-Namensbaum wird zu einer Lookup-Tabelle abgeflacht.

```

147     if not dict then return end
148     local names = pdfc.getarray(dict, "Names")
149     if names then
150         local index = 1
151         while index <= #names do
152             local _, key = pdfc.getfromarray(names, index)
153             local _, item = pdfc.getfromarray(names, index + 1)
154             if key ~= nil and item ~= nil then target[tostring(key)] = item end
155             index = index + 2
156         end
157     end
158     local kids = pdfc.getarray(dict, "Kids")
159     if kids then
160         for index = 1, #kids do
161             local _, kid = pdfc.getfromarray(kids, index)
162             name_tree(as_dict(kid), target)
163         end
164     end
165 end

```

```

166
167 local function named_destinations(doc)
168     local result = {}
169     local names = pdf.getdictionary(doc.Catalog, "Names")
170     if names then name_tree(pdf.getdictionary(names, "Dests"), result) end

```

Für PDF 1.1 kann /Dests ein Katalog-Dictionary sein.

```

171     local legacy = pdf.getdictionary(doc.Catalog, "Dests")
172     if legacy then
173         for key, item in pairs(pdf.dictionarystable(legacy)) do
174             result[tostring(key)] = select(2, value(legacy, key)) or item
175         end
176     end
177     return result
178 end
179
180 local function min_page(a, b)
181     if not a or a == 0 then return b end
182     if not b or b == 0 then return a end
183     return math.min(a, b)
184 end
185

```

Seitenermittlung Fehlt /Pg an einem StructElem, bestimmt der ParentTree dessen Verwendungsseiten; die früheste Seite dient als konservatives Navigationsziel.

```

186 local function build_struct_page_map(root)
187     local result = {}
188     local parent_tree = pdf.getdictionary(root, "ParentTree")
189     if not parent_tree then return result end
190
191     local function register_array(key, array)
192         local page = tonumber(key)
193         if not page then return end
194         for index = 1, #array do
195             local _, item = pdf.getfromarray(array, index)
196             local object_number = ref_number(item)
197             if object_number then result[object_number] =
198                 ↪ min_page(result[object_number], page) end
198         end
199     end
200
201     local walk_number_tree
202     walk_number_tree = function(dict)
203         local _, nums = value(dict, "Nums")

```

```

204     local nums_array = as_array(nums)
205     if nums_array then
206         local index = 1
207         while index <= #nums_array do
208             local _, key = pdf.getfromarray(nums_array, index)
209             local _, val = pdf.getfromarray(nums_array, index + 1)
210             local array = as_array(val)
211             if array then register_array(key, array) end
212             index = index + 2
213         end
214     end
215     local _, kids = value(dict, "Kids")
216     local kids_array = as_array(kids)
217     if kids_array then
218         for index = 1, #kids_array do
219             local _, kid = pdf.getfromarray(kids_array, index)
220             local kid_dict = as_dict(kid)
221             if kid_dict then walk_number_tree(kid_dict) end
222         end
223     end
224 end
225
226 walk_number_tree(parent_tree)
227 return result
228 end
229

```

RoleMap-Übernahme Eigene Rollennamen der Quellstruktur sind ohne ihre RoleMap-Eintragung semantisch unvollständig; Konsumenten der IR (z.B. tagpaxinclude) müssten sie sonst als unbekannt behandeln.

```

230 local function extract_role_map(root)
231     local result = {}
232     local rolemap = pdf.getdictionary(root, "RoleMap")
233     if not rolemap then return result end
234     local keys = {}
235     for key in pairs(pdf.dictionarytotable(rolemap)) do
236         keys[#keys + 1] = key
237     end
238     table.sort(keys, function(a, b) return tostring(a) < tostring(b) end)
239     for _, key in ipairs(keys) do
240         local target = pdf_name(rolemap, key)
241         if target then result[#result + 1] = { tag = tostring(key), role = target }
242         ↪ end
243     end
244 end

```

```

243     return result
244 end
245

```

M.is_tagged prüft nur auf StructTreeRoot und liefert bei ungetaggten PDFs false. Unlesbare oder beschädigte Dateien bleiben Fehler.

```

246 function M.is_tagged(filename)
247     assert(type(filename) == "string" and filename ~= "", "missing PDF filename")
248     local doc = assert(pdfe.open(filename), "cannot open PDF: " .. filename)
249     local root = pdfe.getdictionary(doc.Catalog, "StructTreeRoot")
250     pdfe.close(doc)
251     return root ~= nil
252 end
253
254 function M.extract(filename, outname)

```

Die Extraktion öffnet die Quelle, serialisiert einen kanonischen semantischen Stand und schließt beide Handles als eine Transaktion.

```

255     assert(type(filename) == "string" and filename ~= "", "missing PDF filename")
256     outname = outname or filename:gsub("%.pdf$", "") .. ".tagpax"
257
258     local doc = assert(pdfe.open(filename), "cannot open PDF: " .. filename)
259     local pmap, npages = page_map(doc)
260     local named_dests = named_destinations(doc)
261     local root = pdfe.getdictionary(doc.Catalog, "StructTreeRoot")
262     assert(root, "PDF has no StructTreeRoot")
263     local struct_page = build_struct_page_map(root)
264     local role_map = extract_role_map(root)
265
266     local f = assert(io.open(outname, "wb"))
267     write_record(f, "tagpax", { __order = { "version", "generator" }, version = 1,
268     ↪ generator = M.version })
269     write_record(f, "source", { __order = { "file", "pages" }, file = filename,
270     ↪ pages = npages })
271     for _, entry in ipairs(role_map) do
272         write_record(f, "rolemap", { __order = { "tag", "role" }, tag = entry.tag,
273         ↪ role = entry.role })
274     end
275
276     local nextid, seen = 0, {}
277     local headings, node_meta = {}, {}
278     local streams, stream_by_object, nextstream = {}, {}, 0
279     local destinations, destination_by_key, annotations = {}, {}, {}
280     local annotation_by_object = {}

```

Ziele und Annotationen Zieloperanden können Arrays, Referenzen, Dictionaries oder Namen sein.

```

279   local function destination_array(operand)
280     local current = operand
281     local guard = 0
282     while current and guard < 8 do
283       guard = guard + 1
284       local kind = pdfc.type(current)
285       if kind == "pdfc.array" then return current end
286       if kind == "pdfc.reference" then
287         local _, resolved = pdfc.getfromreference(current)
288         current = resolved
289       elseif kind == "pdfc.dictionary" then
290         local _, resolved = value(current, "D")
291         current = resolved
292       elseif type(current) == "string" then
293         current = named_dests[current]
294       else
295         return nil
296       end
297     end
298     return nil
299   end
300
301   local function register_destination(operand, name)

```

IDs sind beitragslokal; der Importer ergänzt den eindeutigen Namensraum.

```

302     local key = name and ("name:" .. name) or ("object:" .. tostring(operand))
303     if destination_by_key[key] then return destination_by_key[key] end
304     local array = destination_array(operand)
305     if not array then return nil end
306     local items = array_values(array)
307     local page = pmap[ref_number(items[1])]
308     if not page then return nil end
309     local id = "d" .. tostring(#destinations + 1)
310     local destination = {
311       id = id, name = name, page = page,
312       view = tostring(items[2] or "Fit"):gsub("^/", ""),
313     }
314     for index = 3, math.min(#items, 6) do
315       if type(items[index]) == "number" then
316         destination["arg" .. tostring(index - 2)] = items[index]
317       end
318     end

```

```

319     destinations[#destinations + 1] = destination
320     destination_by_key[key] = id
321     return id
322 end
323
324 local function extract_navigation()

```

Der vollständige Namensbaum wird registriert, auch Ziele ohne Quelllink.

```

325     local names = {}
326     for name in pairs(named_dests) do names[#names + 1] = name end
327     table.sort(names)
328     for _, name in ipairs(names) do register_destination(name, name) end
329
330     for page_number = 1, npages do
331         local page = pdf.getpage(doc, page_number)

```

Seiteneigenschaften nutzen den nullbasierten Containerzugriff; pdf.getfromarray für Zielarrays ist einsbasiert.

```

332     local annots = page.Annots
333     if annots then
334         for index = 0, #annots - 1 do
335             local annot = pdf.getdictionary(annots, index)
336             local _, annot_item = pdf.getfromarray(annots, index + 1)
337             if annot and pdf_name(annot, "Subtype") == "Link" then
338                 local rect = array_values(pdf.getarray(annot, "Rect"))
339                 local action = pdf.getdictionary(annot, "A")
340                 local action_type = action and pdf_name(action, "S")
341                 local _, direct_dest = value(annot, "Dest")
342                 local action_dest
343                 if action then action_dest = select(2, value(action, "D")) end
344                 local record
345                 if direct_dest or action_type == "GoTo" then
346                     local operand = direct_dest or action_dest
347                     local destination
348                     if type(operand) == "string" then
349                         destination = register_destination(operand, operand)
350                     elseif operand then
351                         destination = register_destination(operand)
352                     end
353                     if destination then
354                         record = {
355                             action = "GoTo",
356                             destination = destination,
357                         }

```

```

358     end
359   elseif action_type == "URI" then
360     local uri = pdf_string(action, "URI")
361     if uri then record = { action = "URI", uri = uri } end
362   elseif action_type == "GoToR" then
363     local file = pdf_filespec(action, "F")
364     if file and type(action_dest) == "string" then
365       record = {
366         action = "GoToR", file = file,
367         ["remote-destination"] = action_dest,
368       }
369     elseif file and pdf.type(action_dest) == "pdf.array" then
370       local remote = array_values(action_dest)
371       local view = tostring(remote[2] or "Fit"):gsub("^/", "")
372       local supported_views = {
373         XYZ = true, Fit = true, FitH = true, FitV = true,
374         FitR = true, FitB = true, FitBH = true, FitBV = true,
375       }
376       if not supported_views[view] then view = "Fit" end
377       record = {
378         action = "GoToR", file = file,
379         ["remote-page"] = tonumber(remote[1]),
380         ["remote-view"] = view,
381       }
382     end
383   end
384   if record and #rect >= 4 then
385     record.id = "a" .. tostring(#annotations + 1)
386     record.page = page_number
387     record.subtype = "Link"
388     record.llx, record.lly = rect[1], rect[2]
389     record.urx, record.ury = rect[3], rect[4]

```

Diese Schlüssel werden von tagpax nur geschrieben, wenn ein Link von einer logisch identifizierten externen Einheit (z.B. via `osglectures \olref`) stammt; sie erlauben es einem Konsumenten der IR (z.B. der `osglecture`-Integrationsschicht), eine `GoToR`-Annotation nachträglich als projektintern zu erkennen und in einen internen Sprung umzuschreiben, ohne dass tagpax selbst irgendetwas über `osglecture`-Semantik wissen muss.

```

390     local osg_unit = pdf_string(annot, "OSGLectureUnit")
391     if osg_unit then
392       record["osglecture-unit"] = osg_unit
393       record["osglecture-doctype"] = pdf_string(annot,
394         ↪ "OSGLectureType")
395       record["osglecture-lang"] = pdf_string(annot, "OSGLectureLang")

```

```

395         record["osglecture-source-page"] =
396             tonumber(pdf_string(annot, "OSGLectureSourcePage"))
397     end
398     annotations[#annotations + 1] = record
399     local object_number = ref_number(annot_item)
400     if object_number then annotation_by_object[object_number] = record
        ↪ end
401     end
402 end
403 end
404 end
405 end
406 end
407
408 extract_navigation()
409

```

Inhaltsstream-Inventar MCIDs sind streamlokal. Seitenstreams und explizite /Stm-Objekte erhalten daher IDs, bevor MCR-Datensätze sie referenzieren.

```

410 local function emit_stream(id, kind, page, object_number, structparents,
    ↪ subtype)
411     if streams[id] then return id end
412     streams[id] = true
413     write_record(f, "stream", {
414         __order = { "id", "kind", "page", "source-object", "structparents",
            ↪ "subtype" },
415         id = id, kind = kind, page = page, ["source-object"] = object_number,
416         structparents = structparents, subtype = subtype,
417     })
418     return id
419 end
420
421 local function page_stream(page)
422     local id = "p" .. tostring(page or 0)
423     return emit_stream(id, "page", page or 0)
424 end
425
426 local function object_stream(stm, page)
427     local object_number = ref_number(stm)
428     if object_number and stream_by_object[object_number] then return
        ↪ stream_by_object[object_number] end
429     nextstream = nextstream + 1
430     local id = "s" .. tostring(nextstream)
431     local dict = as_dict(stm)

```

```

432     local structparents, subtype
433     if dict then
434         structparents = select(2, value(dict, "StructParents"))
435         subtype = pdf_name(dict, "Subtype")
436     end
437     if object_number then stream_by_object[object_number] = id end
438     return emit_stream(id, "object", page or 0, object_number, structparents,
439         ↪ subtype)
440 end
441
442 local function newid()
443     nextid = nextid + 1
444     return "n" .. nextid
445 end
446
447 local function page_from_ref(ref, fallback)
448     if ref then return pmap[ref_number(ref)] or fallback end
449     return fallback
450 end
451
452 local function emit_mcr(parent, index, dict, inherited_page)

```

MCID und Quell-/K-Index sind semantisch und bleiben erhalten.

```

452     local mcid = select(2, value(dict, "MCID"))
453     if mcid == nil then return nil end
454     local page = page_from_ref(select(2, value(dict, "Pg")), inherited_page)
455     local stm = select(2, value(dict, "Stm"))
456     local stream_id = stm and object_stream(stm, page) or page_stream(page)
457     write_record(f, "kid", {
458         __order = { "parent", "index", "kind", "page", "stream", "mcid" },
459         parent = parent, index = index, kind = "mcr", page = page or 0,
460         stream = stream_id, mcid = mcid,
461     })
462     return page
463 end
464
465 local walk
466 local function walkkid(parent, index, kid, inherited_page)

```

Die polymorphe /K-Grammatik wird in explizite Knoten-, MCR- und OBJR-Kinder normalisiert.

```

467     local kid_type = pdfc.type(kid)
468     if type(kid) == "number" then
469         write_record(f, "kid", {
470             __order = { "parent", "index", "kind", "page", "stream", "mcid" },

```

```

471     parent = parent, index = index, kind = "mcr", page = inherited_page or
      ↪ 0,
472     stream = page_stream(inherited_page), mcid = kid,
473   })
474   return inherited_page
475 elseif kid_type == "pdfe.dictionary" or kid_type == "pdfe.reference" then
476   local dict = as_dict(kid)
477   if not dict then return nil end
478   local typ = pdf_name(dict, "Type")
479   if typ == "OBJR" then
480     local _, object = value(dict, "Obj")
481     local annotation = annotation_by_object[ref_number(object)]
482     if annotation then
483       annotation.parent = parent
484       write_record(f, "kid", {
485         __order = { "parent", "index", "kind", "ref" },
486         parent = parent, index = index, kind = "objr", ref = annotation.id,
487       })
488     end
489     return page_from_ref(select(2, value(dict, "Pg")), inherited_page)
490   end
491   if typ == "MCR" or select(2, value(dict, "MCID")) ~= nil then
492     return emit_mcr(parent, index, dict, inherited_page)
493   end
494   local child_id, child_page = walk(dict, inherited_page, ref_number(kid))
495   if child_id then
496     write_record(f, "kid", {
497       __order = { "parent", "index", "kind", "ref" },
498       parent = parent, index = index, kind = "node", ref = child_id,
499     })
500   end
501   return child_page
502 end
503 end
504
505 walk = function(dict, inherited_page, supplied_object_number)

```

Mehrfach referenzierte indirekte StructElem behalten ihre Graphidentität.

```

506   local object_number = supplied_object_number or ref_number(dict)
507   if object_number and seen[object_number] then
508     local id = seen[object_number]
509     return id, node_meta[id] and node_meta[id].first_page
510   end
511

```

```

512     local id = newid()
513     if object_number then seen[object_number] = id end
514
515     local role = pdf_name(dict, "S") or "Div"
516     local own_page = page_from_ref(select(2, value(dict, "Pg")), inherited_page)
517     own_page = own_page or (object_number and struct_page[object_number])
518     local title = pdf_string(dict, "T")
519     local actual = pdf_string(dict, "ActualText")
520     local alt = pdf_string(dict, "Alt")
521     local lang = pdf_string(dict, "Lang")
522
523     write_record(f, "node", {
524         __order = { "id", "role", "title", "actualtext", "alt", "lang" },
525         id = id, role = role, title = title, actualtext = actual, alt = alt, lang
526         ↪ = lang,
527     })
528
529     local heading_record
530     if role:match("^H[1-6]$") then
531         local text = actual or title or alt
532         heading_record = {
533             node = id, role = role, page = 0,
534             text = text, source = actual and "ActualText" or title and "T" or alt
535             ↪ and "Alt" or "missing",
536         }
537         headings[#headings + 1] = heading_record
538     end
539
540     local first_page = own_page
541     local _, kids = value(dict, "K")
542     local array = as_array(kids)
543     if array then
544         for index = 1, #array do
545             local _, kid = pdf.getfromarray(array, index)
546             first_page = min_page(first_page, walkkid(id, index, kid, own_page))
547         end
548     elseif kids ~= nil then
549         first_page = min_page(first_page, walkkid(id, 1, kids, own_page))
550     end
551
552     node_meta[id] = { first_page = first_page, role = role }
553     if heading_record then heading_record.page = first_page or 0 end
554     return id, first_page
555 end

```

554

Wurzel- und nachgelagerte Datensätze Navigation und Annotationen folgen dem Strukturgraphen, bewahren aber ihre Quellreihenfolge.

```

555     local _, root_kids = value(root, "K")
556     local roots, array = {}, as_array(root_kids)
557     if array then
558         for index = 1, #array do
559             local _, kid = pdf.getfromarray(array, index)
560             local dict = as_dict(kid)
561             if dict then roots[#roots + 1] = (walk(dict, nil, ref_number(kid))) end
562         end
563     else
564         local dict = as_dict(root_kids)
565         if dict then roots[1] = (walk(dict, nil, ref_number(root_kids))) end
566     end
567
568     for index, id in ipairs(roots) do
569         write_record(f, "root", { __order = { "index", "node" }, index = index, node
570             ↪ = id })
571     end
572     for _, heading in ipairs(headings) do
573         write_record(f, "heading", {
574             __order = { "node", "role", "page", "text", "source" },
575             node = heading.node, role = heading.role, page = heading.page,
576             text = heading.text, source = heading.source,
577         })
578     end
579     for _, destination in ipairs(destinations) do
580         write_record(f, "destination", {
581             __order = { "id", "name", "page", "view", "arg1", "arg2", "arg3", "arg4"
582                 ↪ },
583             id = destination.id, name = destination.name, page = destination.page,
584             view = destination.view, arg1 = destination.arg1, arg2 = destination.arg2,
585             arg3 = destination.arg3, arg4 = destination.arg4,
586         })
587     end
588     for _, annotation in ipairs(annotations) do
589         write_record(f, "annotation", {
590             __order = {
591                 "id", "page", "subtype", "action", "destination", "uri", "file",
592                 "remote-destination", "remote-page", "remote-view", "parent",
593                 "llx", "lly", "urx", "ury",
594                 "osglecture-unit", "osglecture-doctype", "osglecture-lang",

```

```

593     "osglecture-source-page",
594 },
595 id = annotation.id, page = annotation.page, subtype = annotation.subtype,
596 action = annotation.action, destination = annotation.destination,
597 uri = annotation.uri, file = annotation.file,
598 ["remote-destination"] = annotation["remote-destination"],
599 ["remote-page"] = annotation["remote-page"],
600 ["remote-view"] = annotation["remote-view"],
601 parent = annotation.parent,
602 llx = annotation.llx, lly = annotation.lly,
603 urx = annotation.urx, ury = annotation.ury,
604 ["osglecture-unit"] = annotation["osglecture-unit"],
605 ["osglecture-doctype"] = annotation["osglecture-doctype"],
606 ["osglecture-lang"] = annotation["osglecture-lang"],
607 ["osglecture-source-page"] = annotation["osglecture-source-page"],
608 })
609 end
610
611 f:close()
612 pdfc.close(doc)
613 return outname
614 end
615

```

Das leere optionale Argument von `\tagpaxextract` kommt als Leerstring an. Die Umwandlung zu `nil` bleibt in Lua, damit der `\directlua`-Aufruf trotz der von `\ExplSyntaxOn` geänderten Leerzeichen-Catcodes ein einzelner Ausdruck bleibt.

```

616 function M.extract_command(filename, outname)
617     return M.extract(filename, outname ~= "" and outname or nil)
618 end
619

```

Kompatibilitätsfassade Lesen und Validieren liegen in eigenen Modulen; diese Funktionen leiten ohne doppelte Implementierung weiter.

```

620 function M.read(filename)
621     return ir_reader.read(filename)
622 end
623
624 function M.validate(ir)
625     return validator.validate(ir)
626 end
627

```

TeX-Navigationsadapter Nur validierte Heading-Datensätze überqueren die Grenze von Lua zu TeX-Makroargumenten. Das Escaping selbst stammt aus `tagpax-ir.lua` (dort auch mit

Begründung, warum es alle relevanten Zeichen abdecken muss); hier kommt nur noch die Normalisierung eingebetteter Zeilenumbrüche hinzu, die innerhalb eines einzelnen T_EX-Arguments nicht zulässig sind.

```

628 local function tex_escape(s)
629     return (ir_reader.tex_escape(s):gsub("\\r?\\n", " "))
630 end
631
632 function M.emit_tex_headings(filename)
633     local ir = M.read(filename)
634     local ok, errors = M.validate(ir)
635     if not ok then error("invalid tagpax IR: " .. table.concat(errors, "; ")) end
636     for _, heading in ipairs(ir.headings) do
637         if heading.text and heading.text ~= "" then
638             tex.sprint(string.format(
639                 "\\tagpax_heading_record:nnnn{%s}{%s}{%s}{%s}",
640                 heading.role, heading.page, heading.node, tex_escape(heading.text)
641             ))
642         end
643     end
644 end
645
646 return M

```

Index

B

<code>booklet</code>	3
<code>bookmark-depth</code>	3

C

<code>CTAN</code>	1 f.
-------------------------	------

D

<code>\directlua</code>	42
<code>\DocumentMetadata</code>	2

E

<code>\ExplSyntaxOn</code>	42
----------------------------------	----

F

FISCHER, Ulrike	2
-----------------------	---

H

<code>heading-map</code>	3
--------------------------------	---

I

<code>\includepdf</code>	3 f.
<code>ir</code>	4

L

<code>\l_tagpax_grid_cellheight_dim</code>	21, 23
<code>\l_tagpax_grid_cellwidth_dim</code>	21
<code>\l_tagpax_grid_cols_int</code>	21
<code>\l_tagpax_grid_rows_int</code>	21
<code>layout</code>	4

M

MATTHIAS, Andreas	1
<code>\MetaDocumentdata</code>	2

N

<code>newpax</code> (Paket)	2
<code>\Note</code>	4
<code>nup</code>	3

O

OBERDIEK, Heiko	2
<code>\olref</code>	36

P

<code>pagecommand</code>	3
<code>pages</code>	3
<code>pax</code> (Paket)	2
<code>pdfpages</code>	4
<code>pdfpages</code> (Paket)	1 ff., 12

S

<code>signature</code>	3
------------------------------	---

T

<code>tagpax</code> (Paket)	1, 3, 11 f.
<code>\tagpaxextract</code>	2 f., 42
<code>\TagPaxGridPageBegin</code>	21
<code>\TagPaxImportOnePage</code>	21
<code>\tagpaxinclude</code>	2 ff., 12, 25
<code>\tagpaxlastprefix</code>	4
<code>\tagpaxsetup</code>	4
<code>tagpdf</code> (Paket)	13
<code>\tagpdfsetup</code>	25
<code>\textheight</code>	21
<code>\thispagestyle</code>	4
<code>toc-depth</code>	3
<code>\topskip</code>	21