Kuonen.PDFTools — Roadmap

Farb-Legende (Status):

✓ fertig	◑ teilweise	○ offen	★ neu / Baustelle	✗ entfaellt / nie

Diese Datei fuehrt nur noch das OFFENE. Alles Erledigte steht nach Version in eigenen

Dateien, damit die Restliste lesbar bleibt:

v4 erledigt · v3 ·

v2 · v1.

Die Spalte offen zaehlt die Zeilen mit oder in dieser Datei -

sie ist damit nachrechenbar und kann nicht mehr veralten.

Stand: aktueller Code M1436. Git-versionierte Kontroll-Roadmap; das Download-PDF (/download/roadmap) wird

ab M1188 direkt aus dieser Datei erzeugt. Alles steht als Tabelle, kaum Fliesstext.

Aufwand: S klein · M mittel · L gross · XL sehr gross.

Status: Fertig · Teilweise · Offen · Zurueckgestellt · Neu/Baustelle · x Nie im PDF.

Aufbau: v4 = geplante Werkzeug-Paritaet (Acrobat/InDesign/Word/Excel/PowerPoint/FOP/LaTeX/HTML/Dreamweaver/Notepad + Bildformate/Schriften/Editor) + externe Punkte · v3 = abgeschlossene grosse Tracks · v2 = History · v1 = Archiv je Bereich.

Verbindliche Wahrheit je Meilenstein: das Ledger in CLAUDE.md.

v4 · Werkzeug-Paritaet - geplant (offen)

Ehrliche Lueckenanalyse im Code geprueft (nicht aus dem Gedaechtnis): was Editor/Bibliothek gegenueber den

grossen Vorbildern noch WENIGER koennen. Alles baubar (Grundprinzip: keine „bauen wir nie"-Grenzen).

Bereits Gebautes ist bewusst NICHT gelistet (z. B. verschachtelte/GREP-Formate, Mehrspaltensatz, Silbentrennung,

Index/TOC/Querverweise/Fussnoten, Data Merge, IDML — schon da). Gruppen: AP Acrobat ·

LXL Excel · LFO FOP · LTX LaTeX · LHT HTML · LPP PowerPoint ·

IMG Bildformate · FNT Schriften · TXT Notepad/Klartext · AG Apogee-Rest · ED Editor-Paritaet · PRN virtueller Drucker · LR Lightroom · LVS Visio · ARC Archive.

v4 auf einen Blick

Zwei verschiedene Zahlen — nicht verwechseln. „Punkte" zaehlt nur die Restliste unten (v4 listet ausschliesslich Luecken; alles Gebaute steht in v1–v3). „Abdeckung" ist die grobe Einschaetzung, wie weit das Werkzeug insgesamt ersetzt ist — eine Gruppe kann 0 erledigte Punkte haben und trotzdem 85 % abdecken.

Grp	Vorbild	n	offen	Abdeckung (grob)	Groesste Luecke
AP	Acrobat Pro	7	0	~97 %	alle Punkte angegangen; Reste sind in den Zeilen benannt
LID	InDesign	6	0	100 %	Fertig (M1430). Zuletzt die hilfslinienbasierte Liquid-Layout-Regel: Hilfslinien sind jetzt Teil des Seitenmodells (je Seite, im .kdoc), fluessige Linien sind Naehte, die die Groessenaenderung aufnehmen
LWD	Word	6	0	100 %	Fertig (M1431, Grenzen geschlossen M1432). Zuletzt Formen, Textrahmen, Gruppen, SmartArt und eingebettete Diagramme — gezeichnet statt uebergangen, ueber das neue gemeinsame Modul DrawingMl (auch von PowerPoint und Excel benutzt)
LXL	Excel	6	0	~92 %	Diagramme/Bilder schreiben
LFO	Apache FOP	5	0	100 %	Fertig. LFO1–LFO5 komplett (goldener Standard)
LTX	LaTeX	6	0	100 %	Fertig (M1428). Die zwei letzten Zeilen sind geschlossen: pgfplots traegt eigene Achsengrenzen und Teilstrichlisten, parametrische Plots, Balken- und Fehlerbalken-Diagramme mit Markern, die Baeume grow= in Grad samt eigenen Abstaenden und den rechtwinkligen Kantenverlauf, und ein eigener edge from parent path wird ueber den vorhandenen Pfadleser ausgewertet statt gemeldet. Dazu der .bst-Interpreter — die vier Originalstile aus TeX Live laufen durch und treffen ein echtes BibTeX 0.99d
LHT	HTML/CSS	6	2	~95 %	nur Medium-Grenzen (Events/Timer im statischen PDF)
LPP	PowerPoint	9	1	~98 %	M1420 hat die ganze Schreib-Seite gebaut (Layouts/Platzhalter, alle prstGeom-Formen mit Verlauf+Schatten, Aufzaehlungen mit wirksamen Tabulatoren, Diagramme). Offen: SmartArt — eigenes Layout-Datenmodell, wird erkannt und gemeldet statt geraten

Grp	Vorbild	n	offen	Abdeckung (grob)	Groesste Luecke
LVS	Visio	24	1	~97 %	M1429 Kern, M1434 Geometrie-Vererbung/Stil/Text/Gruppen/Bildexport, M1435 Kurven/Gruppen/Ebenen/Klebepunkte/.vdx/.vsd 11/Feinabgleich, M1436 eingebettete Bilder, Fuellmuster/Verlaeufe/Transparenz/Schatten, Themen-Varianten und Ebenen im Objektmodell samt Editor-Palette, .vsd der Fassungen 5/6 und der Zeichnungsmassstab. Offen (LVS18): die zur Laufzeit gemeldeten Feinheiten
ARC	Archive (ZIP/RAR/7z/TAR/ISO/I	34	18	~90 %	M1423–M1433 haben alle Formate lesbar gemacht und die meisten schreibbar. Offen: mehrteilige/geteilte Archive (oeffnen und aufteilen), die fehlenden Packer (BZIP2, LZX, UDF, Apple Archive, StuffIt, .dmg), verschluesselte RAR/APFS, El Torito und ein stromweiser Weg fuer sehr grosse Archive. RAR schreiben bleibt aus Lizenzgruenden ausgeschlossen
IMG	Bildformate	33 (davon 6 als Gruppen mit 50 Unterpunkten)	14	Lesen ~99 % · Schreiben ~96 %	AVIF liefert seit M1413 Bildpunkte (IMG6 8/8, bit-genau gegen dav1d) — damit ist von den XL-Codecs nur noch das Schreiben offen (IMG7); ehrlich benannt: .avif bleibt trotzdem aus DocumentConverter draussen, weil libavif-Dateien die Quantisierungsmatrizen brauchen. Restscheiben in IMG7/23/29; IMG8 und IMG32 sind vollstaendig
FNT	Schriften	7	0	Satz ~98 % · Datei-Ausgabe ~90 %	komplett — FNT4 (voller Befehlssatz + Rasterizer-Anbindung, opt-in) und FNT7 (bereits M1316 gebaut, hier nur nachgezogen)
FED	Schrift-Editor	20	0	Lesen ~97 % · Schreiben ~100 %	20/20 fertig, Format-Matrix komplett. FED18 (TTX) geschlossen — zusammengesetzte Glyphen, die CFF -Umriss-Tabelle und die post-Namen; dazu die zwei letzten Matrix-Luecken: Bitmap-Emoji schreiben (ColorBitmapFontEncoder: CBDT/CBLC + sbix) und variables CFF2 schreiben (Cff2Writer, das CFF-Gegenstueck zu FED9)
LIC	Lizenz-Manager	7	0	100 %	Fertig (M1427). Die Lizenz benennt jetzt Gruppen und einzelne Werkzeuge, eine Stelle entscheidet, die Aussteller schneiden mit derselben Auswahl zu und lehnen einen Tippfehler ab statt eine tote Lizenz auszustellen, und der Pruefstand folgt derselben Lizenz. Der Wortschatz liegt als Licensing/LicenseCatalog im Kern — nicht im MCP-Server, den die Aussteller gar nicht referenzieren
FS	Datei- und Ordner-Werkzeuge	8	0	100 %	Fertig. M1424 hat die Gruppe gebaut (sieben Werkzeuge, der geteilte Lauf, die Doppel-Schranke vor den aendernden Punkten); M1425 hat den FS6-Rest geschlossen — Store-Paket-Verweis, Ressourcen-ID und die ausgelagerten .mun-Ressourcen. Es bleiben nur noch Grenzen, die Windows selbst setzt (Signatur-Pruefung, SACL, Unix-Besitzer nach Namen)

Grp	Vorbild	n	offen	Abdeckung (grob)	Groesste Luecke
FCR	Schrift-Erzeuger	16	16	0 %	Neu, auf Nutzerwunsch: ein echter Font-Creator/-Editor in der Darstellung des grossen Editors. Die Rechenkerne liegen aus FED vor (Umrisse lesen/aendern/schreiben, Schrift von Grund auf bauen, Metriken, Kerning, variable Achsen, Farbschriften) — was fehlt, ist das Werkzeug: heute gibt es nur eine Formularseite, auf der sich EIN Umrisspunkt ziehen laesst
MUS	Notensatz	15 (davon MUS13 als Gruppe mit 14 Unterpunkten)	12	Fundament ~70 %	Modell/Stich/PDF/MIDI stehen, seit M1421 Notentext/Gesangstext (MUS5/MUS6) und Editor, seit M1426 Notenschrift Bravura (SIL OFL 1.1) gebuendelt (MUS2) und die feinen Stichregeln vollstaendig (MUS3: Balkenraster Sit/Straddle/Hang, Neigungskontur, Kollisionsausweichen fuer Boegen/Beischriften/Fermaten, elastische Vorzeichen- und Silbenabstaende). Groesste verbleibende Luecke ist OMR (MUS16)
TXT	Notepad	6	0	~98 %	komplett; C#-Hervorhebung ohne Format-Angabe im Interpolationsausdruck
AG	Apogee	4	0	~98 %	komplett
ED	Editor (alle Vorbilder)	24	1	InDesign ~80 % · Acrobat ~30 % · Word ~35 % · Excel ~5 % · PowerPoint 0 % · Dreamweaver ~85 %	Dienste sind da, die Oberflaeche fehlt
LR	Lightroom	14	10	~10 % (Farb-/Bild-Unterbau da)	Entwicklung und Katalog fehlen
PRN	virtueller Drucker	8 (davon PRN3 als Gruppe mit 7 Unterpunkten)	2	~95 %	alle acht Punkte gebaut, und die benannten Grenzen der ersten Runde geschlossen (M1391): PCL mit Delta-Row-Kompression, Symbolsaetzen, Raendern und eingebettetem HP-GL/2; PWG mit CMYK, 16 Bit und CIELab; der Auftrag traegt seine echte Herkunft; Cancel-Job greift ueber den zweiphasigen Create-Job/Send-Document-Weg; das Kontingent uebersteht einen Neustart; Anmeldung und ipps://. Es bleiben nur noch Grenzen, die die Maschine setzt (kein Windows-Drucksystem, kein GhostPCL zum Gegenlesen) Seit M1409 auch PRN3 vollstaendig (7/7): mDNS mit Sondieren/Konflikt/Abmelden, WSD als Windows-Weg, IPP Everywhere, und der gemessene Netzweg. Was der Code kann, ist damit gebaut; was fehlt, ist der Nachweis am fremden Geraet — abgenommen ist gegen echtes CUPS (treiberlose Einrichtung + zwei angekommene Auftraege), ungeprueft bleiben Windows, macOS, iOS und Android (kein zweites Geraet im Netz). Dazu zwei benannte Luecken: WS-Print (die WSD-Druckseite) ist nicht gebaut, und auf einem Windows-/macOS-Wirt traegt keine Docker-Betriebsart die Bekanntmachung ins LAN — dort laeuft sie ueber den wirtsseitigen Ankuendiger.

Grp	Vorbild	n	offen	Abdeckung (grob)	Groesste Luecke
EXT	extern	4	0	—	toolchain-/zertifikatsgebunden, kein fehlender Code

Dreamweaver hat keine eigene Gruppe: der HTML/CSS->PDF-Weg ist ~95 % (LHT), die Oberflaeche (Code-Editor mit Live-Vorschau /htmleditor, Site-Verwaltung, HTML-Vervollstaendigung/-Validierung) steht seit ED50/ED51 (M1372-Nachtrag); offen bleibt nur die CSS-Haelfte von ED52.

MG · Gemeldete Maengel an bereits Gebautem

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

Reihenfolge, die ich empfehle

Nicht nach Aufwand sortiert, sondern nach Nutzen je Aufwand.

Stand M1427 (Rang 1 „ARC7 TAR-Ketten" ist erledigt und herausgenommen; FCR1–FCR4 rueckt nach Rang 1).

Rang	Was	Warum zuerst	Aufw.
1	FCR1 – FCR4 die Huelle, die Leinwand und das Zeichenwerkzeug	Die tragende Scheibe des Schrift-Editors: erst damit wird aus dem Betrachter ein Werkzeug. Alle Rechenkerne stehen (FED 20/20), es fehlt die Bedienung	L
2	LR Fotoentwicklung und Katalog	Zehn offene Punkte; der Rechenkern der Entwicklung steht seit M1404, Katalog und Oberflaeche fehlen	XL
3	IMG7 die XL-Codecs schreiben	Der letzte grosse Bildpunkt. AVIF liefert seit M1413 bit-genaue Bildpunkte — offen ist die Gegenrichtung	XL

Nicht in der Liste, mit Absicht: LHT3/LHT4, LPP8 (SmartArt) und EXT sind entweder Nischen, Medium-Grenzen (Bewegung in einem statischen PDF) oder nicht abbildbar — sie stehen in ihren Zeilen mit Begruendung und melden sich zur Laufzeit.

v4-AP · Acrobat Pro

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

v4-LXL · Excel (.xlsx)

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

v4-LFO · Apache FOP (XSL-FO)

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

v4-LHT · HTML/CSS (Browser-Rendering)

Schon da: toleranter Leser, CSS-Kaskade + erweiterte Selektoren, Boxmodell, Float, position (relative/absolute/fixed), Mehrspalten (column-count), Tabellen (colspan/rowspan), Inline-SVG, Web-Fonts (opt-in Net-Add-on), JS-Subset (opt-in).

#	Browser-Funktion	Stand bei dir	Aufw.
LHT3	CSS Transforms (translate/rotate/scale + transform-origin)	M656/M886; Transitions/Animations = statisches Standbild (Medium-Grenze)	M
LHT4	JavaScript - Ladezeit-Subset + skript-definierte Funktionen/Closures/Rekursion M1290; Events/Timer/fetch sind in einem statischen PDF nicht ausfuehrbar (Medium-Grenze)	M1290	XL

v4-LPP · PowerPoint (.pptx)

Schon da: positionierte Text-/Bildrahmen lesen+schreiben (Folienmasse, Absaetze/Laeufe mit Fett/Kursiv/Groesse/Farbe/Ausrichtung, Bilder aus ppt/media), pptx->pdf, pdf->pptx (M1205); dazu seit LPP5-LPP9 Notizen/Kommentare, Themenfarben+-schrift, Bild-Zuschnitt/Gruppen/Drehung/Spiegelung, lesende Diagramme (Balken/Linie/Torte) und der rohe Uebergangs-/Zeitplan-Rundlauf. Der Leser kennt heute sp/pic/grpSp/graphicFrame+c:chart/txBody — Layouts/Master-Platzhalter, Tabellen und Formen-Geometrie (prstGeom) sind unten im Code geprueft NICHT vorhanden (LPP1-4).

#	PowerPoint-Funktion	Stand bei dir	Aufw.
LPP8	Diagramme + SmartArt (c:chart, DrawingML-Diagramme)	M1420: Diagramme (Balken/Linie/Torte) jetzt auch geschrieben (c:chart-Teil mit Beziehung und Inhaltstyp), nicht mehr nur gelesen. SmartArt offen: eigenes Layout-Datenmodell (dgm:relIds) - wird erkannt und gemeldet (PPTX_SMARTART_NOT_RENDERED) statt geraten. Dabei aufgefallen: Tabellen fielen bisher STILL aus dem Rundlauf, sie werden jetzt ebenfalls gemeldet	L (Nische)

v4-IMG · Bildformate — Lesen und Schreiben

Schon da beim Lesen (je eigener Codec, kein Fremdpaket): JPEG, PNG, TIFF (unkomprimiert/LZW/Deflate/PackBits/
CCITT G3+G4/JPEG-in-TIFF, 1–16 Bit), GIF, WebP (VP8L und VP8), HEIF/HEIC (eigener HEVC-Intra-Decoder, M1147),
JPEG 2000 (EBCOT/DWT), JBIG2, CCITT.
Geschrieben wurde bei Beginn dieser Gruppe nur PNG (indiziert/TrueColor), JPEG und 1-Bit-TIFF
(unkomprimiert, fuer CTP) — genau das war die Luecke: die Lese-Seite war deutlich breiter als die Schreib-Seite.
Seither ist sie geschlossen (siehe die Matrix); offen sind nur noch die drei XL-Codecs AV1/AVIF, HEIF-Schreiben
und JPEG XL.
Die Matrix — welches Format in welche Richtung (im Code geprueft):

Form	Lesen	Schreib	Bemerkung
AVIF	Bildpunkte, bit-genau gegen dav1d - verlustfrei UND verlustbehaftet (M1310-M1413)	x	nicht allgemein tragbar: libavif (der Kodierer hinter Chrome, avifenc, Pillow, ImageMagick) schaltet bei jeder verlustbehafteten Stufe die Quantisierungsmatrizen ein, die hier nicht hinterlegt sind - deshalb bleibt .avif aus DocumentConverter draussen (sonst genau der IMG1-Fehler). Schreiben braucht einen AV1-Encoder -> IMG7

Kassensturz (aus dem Code gelesen): bei M1306 waren von 15 lesbaren Rasterformaten 6 schreibbar
(PNG, JPEG, TIFF, GIF, WebP verlustfrei, JPEG 2000 verlustfrei). Heute sind es alle bis auf HEIF/AVIF, JPEG XL
und DNG — und DNG ist bewusst nicht vorgesehen (eine Rohdatei zu schreiben hiesse, Sensordaten zu erfinden).
Die folgenden Punkte fuehren die Gruppe zu Ende.

#	Bildformat-Funktion	Stand bei dir	Aufw.
IMG7 6/10	HEIF/AVIF schreiben (HEVC- bzw. AV1-Intra-Encoder)	Untertabelle unten: IMG7.1–7.10	XL
IMG8 4/6	JPEG XL lesen + schreiben	Untertabelle unten: IMG8.1–8.6	XL
IMG23 6/8	JBIG2 schreiben — gescannte Schwarz-Weiss-Seiten deutlich kleiner als CCITT G4	Untertabelle unten: IMG23.1–23.8	L

Detailtiefe je Format. Ein Format „koennen" heisst nicht, jede seiner Spielarten zu koennen. Der
folgende Durchgang ist am Code gemessen und nennt, was je Format noch fehlt — meist Kompressions- oder
Struktur-Varianten, die in echten Dateien vorkommen.

#	Was fehlt	Warum es vorkommt	Aufw.
IMG29 5/7	JPEG: progressiv schreiben; 12 Bit und arithmetische Kodierung lesen	Untertabelle unten: IMG29.1–29.7	M–L

Die grossen Bildformat-Punkte im Einzelnen

Ein Punkt mit mehr als ein paar Arbeitsschritten steht hier als Gruppe mit nummerierten Unterpunkten - sonst verbirgt ein einzelnes genau das, was man wissen will: wie weit es ist und was noch fehlt. Jede Scheibe ist zugleich der Schnitt fuer ein Agenten-Paket.

IMG7 · HEIF/AVIF schreiben — 6 von 10

#	Scheibe	Stand
IMG7.7	Schleifenfilter beim Kodieren (Deblocking + SAO)	M1408 - der SAO-Entscheidungskern waehlt je Baumblock den besten Versatz (vier Kantenklassen, 32 Bandfenster) und ist gegen die vorhandenen HevcSao-Filterkerne belegt: die gewaehlten Werte werden wirklich angewandt und der Fehler davor/danach gemessen. Offen: das Deblocking und die Verdrahtung in den Strom - im Slice-Header bleibt der Filter vorerst abgeschaltet
IMG7.8	Volle Modus- und Quadbaum-Entscheidung (Rate-Distortion)	- heute Summe der absoluten Differenzen und feste Tiefe
IMG7.9	Transform-Skip	- bei Text und Grafik ein grosser Gewinn
IMG7.10	4:4:4 / 4:2:2 und 10 Bit	- gemessen: bei 4:2:0 liegt der Gesamtfehler eines Bildes mit Farbwechsel je Bildpunkt bei 85, bei einem Helligkeitsfehler von 0,4; der Verlust steckt also vollstaendig in der Unterabtastung
—	AVIF schreiben	- braucht einen AV1-Encoder und folgt nach IMG6

IMG8 · JPEG XL — 4 von 6

#	Scheibe	Stand
IMG8.5	VarDCT-Pfad (XYB, variable DCT, Gabor/EPF)	
IMG8.6	Schreiben (modular, verlustfrei)	

IMG23 · JBIG2 schreiben — 6 von 8

#	Scheibe	Stand
IMG23.5	Verfeinerungs-Region schreiben	- legt zwei fast gleiche Buchstaben als ein Symbol plus exakte Differenz ab (bleibt verlustfrei)
IMG23.6	Halbton-Region schreiben	- gerasterte Graustufen ueber ein Musterwoerterbuch

IMG29 · JPEG: die Randfaelle — 5 von 7

#	Scheibe	Stand
IMG29.6	SOF11 — verlustfrei arithmetisch	- bleibt benannt abgelehnt, und der Grund ist gemessen, nicht vermutet (M1408)

#	Scheibe	Stand
IMG29.7	Sukzessive Annaeherung beim progressiven Schreiben	- das Bild baut sich in zwei Stufen auf, nicht in fuenf

IMG32 · JPEG 2000: das volle Codestream-Profil — 11 von 11

v4-FNT · Schriften — alle Arten, beide Richtungen

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

v4-FED · Schriften — jede Art lesen, schreiben und ineinander wandeln, plus Editor

Der Wunsch: moeglichst alle Schriftarten, die es gibt und die benutzt werden — Web, Druck und Betriebssysteme — lesen, schreiben und in jede andere Art wandeln, ohne Fremdprodukt (kein FontForge, kein fonttools). Dazu ein Editor, um bestehende Schriften zu aendern und neue zu bauen.

Was schon da ist (nicht doppelt bauen): Leser fuer TrueType(glyf), OpenType/CFF, bare CFF, CID-keyed CFF, CFF2, Type1(PFB/eexec), Type3, Type42, WOFF, WOFF2, Variable Fonts (inkl. Instanzieren) und Farbfonts COLR/CPAL v0+v1; Umriss-Auslesung (GlyphBoundsReader), sfnt-Schreiber (SfntWriter, SfntTableBuilders), CFF->OpenType-Verpackung, Subsetter fuer TrueType und CFF, Type1FontConverter (-> PFB/PFA), Cff2StaticFontBuilder und FontExporter (M990), der eine eingebettete Schrift bereits als .ttf/.otf/.pfb herausschreibt und fehlende Tabellen ergaenzt.

Die Format-Matrix — wo steht was

Format	Wofuer	Lesen	Schreiben
dfont / Suitcase	macOS	sfnt-Ressourcen (DfontReader, M1364)	FED16 (DfontWriter - nur sfnt-Ressourcen, keine FOND/NFNT)

Die Punkte

Ehrlich zur Verifikation: der ueberzeugende Beleg fuer eine selbst gebaute Schrift ist, dass ein fremder Renderer sie liest (HarfBuzz/FreeType, wie schon bei den OpenType-Attributen M1087). Nach der Projektregel geschieht das einmalig beim Bauen; im Repo bleibt die eingebackene Erwartung.

Ehrlich zur Vollstaendigkeit: „alle Schriftarten" heisst hier alle, die im Web, im Druck und in

Betriebssystemen tatsaechlich vorkommen — die Matrix oben zaehlt sie auf. Firmeneigene Editor-Formate (FontLab VFB/VFC, Glyphs .glyphs) sind nicht dokumentiert; sie liessen sich nur durch Nachbau aus Beispieldateien erschliessen und stehen bewusst nicht in der Liste.

v4-FCR · Schrift-Erzeuger und -Editor (Nutzerwunsch)

„ich moechte einen echten font creator/editor haben mit der darstellung des editors"
Zuerst, was es schon gibt und was nicht — sonst klingt dieser Abschnitt nach einer Luecke, die keine ist. Die Rechenkerne stehen vollstaendig (v4-FED, 20/20): Umrisse lesen, aendern und zurueckschreiben (glyf und CFF-Charstrings), eine Schrift von Grund auf bauen, Metriken und Kerning, cmap, Glyphennamen, variable Achsen (gvar und CFF2), Farbschriften in allen drei Arten, und die Ausgabe in jedes Format der Matrix. Was fehlt, ist das Werkzeug darueber: /schrifteditor (FED7) ist eine Formularseite, auf der sich eine Glyphe waehlen, ein Umrisspunkt ziehen und die Vorbreite setzen laesst. Das ist kein Schrift-Editor, und es sollte auch nie einer sein — es war der Nachweis, dass die Kerne von aussen bedienbar sind.
Der Massstab ist ausdruecklich der grosse Editor, nicht eine Formularseite: Werkzeugleiste, Eigenschaften-Feld an der Seite, freie Arbeitsflaeche mit Linealen, Rueckgaengig-Kette, dasselbe dunkle Thema. Vorbilder sind FontForge, Glyphs und RoboFont.
Die Bauregel aus v4-ED gilt hier von Anfang an (dort wurde sie erst nach 7170 Zeilen Editor.razor erzwungen): Rechenlogik in eigene Playground/Model/-Dateien als reine, testbare Funktionen, Oberflaeche in eigene Komponenten — in der Seite selbst steht nur die Einbindung.

#	Punkt	Stand	Aufw.
FCR1	Die Huelle in der Darstellung des Editors: Werkzeugleiste, Eigenschaften-Feld, Arbeitsflaeche mit Linealen, Rueckgaengig-Kette, Speichern/Laden im Arbeitsbereich — freistehend wie der Dokument- und der Noten-Editor		L
FCR2	Die Glyphen-Leinwand: Em-Quadrat, Grundlinie, x-Hoehe, Versalhoehe, Ober- und Unterlaenge sowie die beiden Seitenraender als Hilfslinien; zoomen, schieben, an den Umriss einpassen		M

#	Punkt	Stand	Aufw.
FCR3	Zeichenwerkzeug: neue Umrisse ziehen, Punkte einfuegen und loeschen, zwischen Ecke / weich / tangential umschalten, Pfad schliessen — der Punkt, ohne den es ein Betrachter bleibt		L
FCR4	Bezier-Griffe mit Stetigkeit: Kontrollpunkte ziehen, und ein weicher Punkt haelt seine Griffe kollinear — sonst entsteht bei jedem Zug ein sichtbarer Knick		M
FCR5	Umriss-Operationen: verschieben, skalieren, drehen, neigen, spiegeln, ausrichten und verteilen, Richtung umkehren (sonst faellt die Fuellung aus), Ueberlappungen entfernen. Ehrlich vorweg: der vorhandene Layout/Pathfinder (LID2) rechnet auf geradkantigen Vielecken — ein Glyphenumriss besteht aus Bezierkurven; das ist ein neuer Algorithmus, keine Wiederverwendung		XL
FCR6	Komponenten: eine Glyphe als Referenz setzen (ä = a + Trema) mit Versatz und Skalierung, Referenz aufloesen, und die Gegenrichtung — aus zwei Umrissen eine Komponente machen. Die Leseseite kann es bereits (FED1/FED18)		M
FCR7	Metriken und Kerning bedienbar: Vor- und Nachbreite, Dickte, Kerning-Paare und -Klassen, dazu das Metrik-Fenster mit einer Wortprobe — Abstaende beurteilt man am Wort, nie an der Einzelglyphe		L
FCR8	Glyphen-Uebersicht: alle Glyphen als Raster, nach Unicode-Bereich gefiltert, Glyphe anlegen / loeschen / umbenennen — und sichtbar, welche Zeichen noch fehlen		M
FCR9	Schrift-Angaben als Dialog: Familien- und Schnittname, PostScript-Name, Version, Hersteller, Lizenzangabe, OS/2-Metriken, Panose. Heute liegen sie nur im Code erreichbar		M
FCR10	Probe und Wasserfall: eigener Text in mehreren Groessen, vorher/nachher nebeneinander, gerastert in echten Bildschirmgroessen — der einzige Weg, eine Aenderung zu beurteilen		M
FCR11	Rueckgaengig-Kette und Zwischenablage: je Glyphe und dokumentweit; Glyphen zwischen zwei Schriften kopieren		M
FCR12	Raster, Fang und Messen: an Hilfslinien, ganzen Einheiten und dem Em-Raster fangen; Abstaende messen und bemassen		M
FCR13	Variable Schriften bearbeiten: Achsen anlegen, Schnitte setzen, den Zwischenschnitt am Schieberegler live sehen. Der Rechenkern steht (FED9 gvar, Cff2Writer) samt seiner benannten Grenze: keine Eck-Master		L
FCR14	Farbschriften bearbeiten: COLR/CPAL-Ebenen anlegen und faerben, Bitmap- und SVG-Glyphen setzen — die Schreibseite steht (FED13, FNT2, FNT3), die Bedienung fehlt		L
FCR15	Zeichnung uebernehmen: einen SVG-Pfad oder ein Vektorzeichen aus einem PDF als Glyphe einsetzen — der uebliche Weg, wie ein Logo zur Schrift wird		M
FCR16	MCP und REST: dieselben Faehigkeiten ohne Oberflaeche, damit ein Generator eine Schrift bauen kann. font_info/font_edit (FED20) decken heute nur Auskunft und Einzelaenderung ab		M

Ehrlich benannt, was hier NICHT hineingehoert. Ein Hint-Editor (Punkte an Zonen binden, Delta-Anweisungen setzen) ist ein eigener Schritt: der Bytecode-Interpreter steht (FNT4, 122 Instruktionen), das Schreiben von Hinting-Programmen von Hand ist etwas anderes. Ebenso das automatische Hinting und die automatische Zeichenabstand-Ermittlung — beides sind eigene Verfahren, keine Oberflaechenarbeit. Und ein Zeichnen mit dem Stift auf dem Tablett setzt

Druck- und Neigungsdaten voraus, die eine Blazor-Server-Oberflaeche nicht bekommt.
Die Reihenfolge ist erzwungen, nicht gewaehlt: FCR1 und FCR2 tragen alles Uebrige — ohne Arbeitsflaeche mit den richtigen Hilfslinien laesst sich kein Punkt sinnvoll setzen. FCR3 und FCR4 gehoeren zusammen (ein Zeichenwerkzeug ohne Stetigkeit erzeugt nur Knicke). FCR13 und FCR14 setzen FCR3 voraus, FCR7 setzt FCR10 voraus.

v4-TXT · Klartext / Notepad

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

v4-AG · Apogee-Rest (Druckvorstufe)

Nichts mehr offen. Die erledigten Punkte samt ihren ehrlich benannten Grenzen stehen in roadmap-erledigt-v4.md.

v4-ED · Editor — Paritaet zu den Vorbildern

Der Editor selbst (nicht die Bibliothek): was die Oberflaeche gegenueber InDesign, Acrobat Pro, Word, Excel,

PowerPoint, Dreamweaver und Notepad noch WENIGER kann. Im Code der Editor-Seite geprueft (Editor.razor 7170 Zeilen + EditorService.cs 4681 Zeilen), nicht aus dem Gedaechtnis.
Schon da (NICHT gelistet): WYSIWYG-Canvas mit Live-Darstellung, Inline-Textbearbeitung, Undo/Redo, Menue-/Kontroll-/Statusleiste, Rechtsklick-Menue, Musterseiten + Doppelseiten + Miniaturen, Lineal-Hilfslinien, Montageflaeche, Ebenen/OCG, Farbfelder mit eigenem Farbwaehler, Verlaeufe, Absatzformate, Verknuepfungs-Palette, Formularfeld-Bedienung, Lesezeichen, Tabellen mit Zelltext/-farben, Drehung, Suchen+Ersetzen, Rechtschreibpruefung auf Knopfdruck, Seiten-Organisierer mit Drag&Drop, Dokument-Import/-Export, PDF-Export-Dialog.

#	Vorbil	Editor-Funktion	Stand bei dir	Aufw
ED43	Word	Fliesstext-Modus: ein durchgehendes Textdokument, bei dem beim Tippen neue Seiten entstehen (heute rahmenbasiert wie InDesign)	Playground/Model/FlowEditorDocument + FlowInlineMarkup + FlowEditorTakeover, Palette „Fliesstext-Modus" NEBEN dem Rahmen-Modus (der bleibt unveraendert). Tippen -> Absaetze (#/-/1./---) -> FlowDocument.Build() -> die Seiten entstehen von selbst. M1381: Auszeichnung einzelner Woerter (fett/kursiv/_unterstrichen_ -> StyledRun), Tabellen (| a | b | mit Kopfzeilen-Trennzeile) und Bilder (![name], ![name|120x60]) im Fluss (ein unbekannter Bildname wird gemeldet), und die Uebernahme in das Rahmen-Dokument ueber denselben PdfToLayoutDocument wie der Datei-Import - damit in der Rueckgaengig-Kette und im .kdoc. Offen: kein Schreibstrich IM Satzbild (Treffer je Glyphe, Auswahl ueber Zeilenumbrueche, Tastaturnavigation im Satz - ein eigener, grosser Schritt), keine Formeln/Fussnoten/Inhaltsverzeichnisse im Fluss; nach der Uebernahme sind es Rahmen, kein Fliesstext mehr	L

v4-LVS · Visio — Rest (die in M1429 benannten Grenzen)

Der Kern ist seit M1429 fertig (siehe roadmap-erledigt-v4.md). Offen ist, was damals ehrlich als Grenze benannt und zur Laufzeit gemeldet wurde.

#	Punkt	Stand bei dir	Aufw.
LVS18	Feinheiten, Rest: Bilder in binaeren .vsd (die Bild-Chunks), Themen-Effekte (Schein/Relief aus der Effekt-Matrix — heute gemeldet), Verlaeufe nach Visio 2013 mit anderer Richtung als linear 0 (heute einfarbig und gemeldet), die Schatten-Unschaerfe im PDF, Schatten und Verlaeufe, die mit der Form drehen, und Stilvorlagen im Zeichnungsmassstab	je zur Laufzeit gemeldet	M

v4-ARC · Archive — Rest

Alle Formate sind seit M1433 lesbar (siehe roadmap-erledigt-v4.md). Offen sind die mehrteiligen

Archive und die Grenzen, die M1433 ehrlich benannt hat.

#	Punkt	Stand bei dir	Aufw
ARC1	Mehrteilige Archive — die Schnittstelle: die Fassade nimmt heute genau EIN byte[]; es braucht eine Form mit mehreren Teilen in fester Reihenfolge, und MCP/Pruefstand finden die Geschwister am Namensmuster (.001, .z01, .part2.rar, …)	fehlt	S
ARC1	7z mehrteilig (.7z.001, .002, …) oeffnen und aufteilen — die Teile sind reine Byte-Stuecke	fehlt (in der Klassendoku als Grenze benannt)	S
ARC1	ZIP geteilt/gespannt (.z01 … .zip) oeffnen und schreiben — Versaetze je Teil, das zentrale Verzeichnis kann ueber Teilgrenzen reichen; Gegenpruefung 7-Zip/Info-ZIP	die Datentraegernummern werden ignoriert, der Schreiber setzt sie fest auf 0	M
ARC1	RAR mehrteilig lesen (.part1.rar, .r00) — ein Eintrag bzw. der solide Strom laeuft ueber die Teilgrenze	die Teile werden erkannt, ein Eintrag ueber die Grenze benannt abgelehnt. Schreiben: nie (Lizenz)	M
ARC1	CAB-Ketten (mehrere Kabinette) lesen und schreiben — ein Ordner-Strom setzt sich im naechsten Kabinett fort; Gegenpruefung makecab	die Verweise auf voriges/naechstes Kabinett werden nicht ausgewertet	M
ARC1	7z verschluesselt schreiben (AES-256 + SHA-256-Schluesselableitung) — das Lesen steht	fehlt	S
ARC2	archive_extract bei soliden 7z/RAR einmal auspacken statt je Eintrag neu	nur Laufzeit, kein Fehler	S
ARC2	ISO mit El Torito (bootfaehige Abbilder) lesen und schreiben	fehlt	S
ARC2	Apple Archive schreiben (blank und im pbz*-Mantel; LZFSE-Packer)	nur gelesen	M
ARC2	BZIP2-Packer (Sortierung aller zyklischen Verschiebungen, Huffman-Tabellenwahl) — dann auch .tar.bz2 und BZIP2 im ZIP schreiben	nur gelesen	M
ARC2	LZX-Packer fuer CAB	CAB wird mit Store/MSZIP geschrieben	M
ARC2	UDF schreiben — Gegenpruefung Linux-Kern und mkudffs	nur gelesen	M
ARC2	RAR verschluesselt lesen (RAR5 AES-256/PBKDF2, RAR4 AES-128) und die Wiederherstellungsdaten	benannt abgelehnt	M
ARC2	StuffIt vervollstaendigen: die Methoden 5/6/8/14, Verschluesselung und StuffIt X lesen, dazu StuffIt schreiben (unar/lsar taugen als Gegenpruefung)	Methoden 0/1/2/3/13/15 lesen	M
ARC2	.dmg schreiben — UDIF-Behaelter mit einem HFS+-Schreiber darin	nur gelesen	L
ARC2	APFS verschluesselt (mit Passwort/Schluessel), versiegelte Volumes, LZBITMAP, bvx1; Fusion als Sonderfall	benannt abgelehnt	L

#	Punkt	Stand bei dir	Aufw
ARC3	Stromweiser Weg fuer Archive, die nicht in den Speicher passen — durch alle Leser und Schreiber	jedes Archiv liegt ganz im Speicher	L
ARC3	StuffIt-Konstanten aus unar (LGPL): die festen Codelaengen der Methode 13 und die Arsenic-Tabelle — rechtliche Einschaetzung durch den Nutzer, sonst unabhaengig herleiten	Entscheidung offen	S

v4-MUS · Notensatz — Notenblaetter erstellen, mit eigenem Editor (Nutzerwunsch)

Ziel: Notenblaetter mit allem, was dazugehoert, selbst setzen — und einen eigenen Editor dafuer.
Vorbilder sind Sibelius/Finale/Dorico und das freie MuseScore; der Austausch laeuft ueber MusicXML (das Format, das alle koennen) und MIDI.
Was dafuer schon da ist (nicht doppelt bauen): das Vektor-Zeichnen (PageCanvas), das Objekt-Modell mit Seiten (LayoutDocument) und sein .kdoc-Format, die Schrift-Einbettung samt Umriss-Auslesung und OpenType-Shaping (Notenschriften wie Bravura sind gewoehnliche OpenType-Schriften nach dem SMuFL-Standard), der abgesicherte XML-Leser (fuer MusicXML) und System.IO.Compression (fuer das gepackte .mxl), dazu die ganze Editor-Oberflaeche (Canvas, Auswahl, Undo/Redo, Paletten), auf der ein Noten-Editor aufsetzen kann.

#	Funktion	Stand bei dir	Aufw
MUS4	Umbruch: Takte auf Systeme, Systeme auf Seiten, Zeilenausgleich, Wiederholungen und Sprungmarken (D.C./D.S./Coda), Taktzahlen	Music/ScoreLayout + ScoreAssembly (M1404): Zeilen- und Seitenumbruch in zwei Durchlaeufen - erst jeden Takt probeweise setzen, um seine natuerliche Breite zu messen (kein zweiter Rechenweg), daraus Spaltenbreiten als Maximum ueber alle Stimmen, sodass die taktweise Ausrichtung zwangslaeufig entsteht; gedehnt wird nur der Abstand zwischen den Zeitstellen, nie die Geometrie einer Note. Die letzte Zeile bleibt ungedehnt (Stichregel). Offen: Wiederholungen und Sprungmarken, Taktzahlen, Seitenkopf/-fuss; der Umbruch ist greedy, kein Knuth-Plass-Optimum	L
MUS7	Mehrere Systeme: Klavier-Akkolade, Chor-/Orchesterpartitur, Systemgruppen und Klammern, Stimmenauszug aus der Partitur, Transposition (klingend notiert)	(M1404) Geschweifte Akkolade und eckige Systemklammer ueber EngravingRules.Groups, durchgehende Taktstriche nur dort, wo alle Systeme der Gruppe einen haben, Stimmennamen links (erste Zeile ausgeschrieben, Folgezeilen Kurzform). Transposition ueber Music/InstrumentTransposition - diatonisch, also Stufenversatz UND Halbtoene: aus klingendem b wird bei der B-Klarinette c'' und nicht his; die Tonart geht durch denselben Rechenweg. Offen: der Stimmenauszug aus der Partitur, und Part.StaffCount > 1 (Klavier/Orgel brauchen zwei Notenzeilen fuer EINE Stimme - heute ist es je Stimme eine)	L

#	Funktion	Stand bei dir	Aufw
MUS8	Sonderfaelle: Schlagzeug-Notation, Gitarren-Tabulatur mit Griffbildern, Prozent-Wiederholungen, Cue-Noten, mehrtaktige Pausen		M
MUS12	Als PDF setzen — ueber den vorhandenen PageCanvas-Weg, kein zweiter Zeichenweg; dazu SVG- und Bild-Ausgabe	Music/MusicPdf.Render (+ DrawSystem, um Noten in ein groesseres Dokument zu setzen): Notenlinien, Haelse, Hilfslinien und Taktstriche als Vektoren, Balken als gefuellte Vierecke, die Zeichen ueber die eingebettete SMuFL-Schrift. Seit M1403 wird der Notenkopf auch ohne Notenschrift gezeichnet (er ist Geometrie, keine Typografie); seit M1404 mehrere Seiten samt Akkolade, Klammern und Stimmennamen. Belegt: 400 Takte auf zwei Systemen ergeben 34 Seiten in rund 110 ms. Offen: die SVG-/Bild-Ausgabe ueber die vorhandenen Wandler	M
MUS13 9/14	Der Noten-Editor — Untertabelle unten: MUS13.1–13.14 (Nutzerwunsch: „so das er freistehend sein kann wie der editor und auch sonst alles naeher am editor")	Pruefstand-Seite /noteneditor mit eigenem Knopf: Spuren mit Instrument (39), Systemgruppen, Notenblatt-Definition, Umschalter klingend/notiert, Vorschau und PDF - seit M1421 dazu die Eingabezeile: klicken setzt, ziehen waehlt aus, Pfeiltasten und Buchstaben setzen und verschieben, dazu Werkzeugleiste, Rueckgaengig-Kette und Speichern/Laden. Die Logik liegt in reinen, testbaren Funktionen unter Playground/Model/, die Razor-Seite ruft nur auf. Offen: Akkorde und mehrere Stimmen je Takt in der Oberflaeche (der Notenstich kann beides), Dynamik-/Artikulations-Paletten, Haltebogen von Hand, Taktmass- und Tonartwechsel mittendrin	XL
MUS14	Anhoeren: die Partitur als Klang wiedergeben (MIDI-Ausgabe an das System bzw. ein einfacher eigener Synthesizer fuer die Vorschau)		L
MUS16	Notenblatt-Erkennung (OMR): aus dem Bild eines gedruckten Notenblatts zurueck zum Score	Neues Modul MusicOcr (M1404): Notenzeilen finden (der Zeilenabstand ist der Massstab, an dem in einem Notenblatt alles haengt - nie Bildpunkte), Notenlinien entfernen ohne die Zeichen zu zerreissen, Notenkoepfe (gefuellt/offen), Haelse, Balken, Punktierung, Taktstriche. Eine Eingangsstelle fuer PDF (jede Seite) und jedes Bildformat ueber RasterImage.Parse. Seit dem ersten echten Pruefblatt: quer liegende Scans werden erkannt und die Drehung gemeldet; dabei fiel ein Fehler im geteilten OcrImageRotation auf - eine Vierteldrehung ueber Winkelfunktionen laesst zwei Bildpunkte duenne Linien zerfallen (0 statt 6 gefundene Notenzeilen), sie wird jetzt durch Umsortieren gerechnet. Ehrlich benannt: Schluessel, Tonart und Taktart werden angenommen, nicht gelesen; Vorzeichen und Pausen werden gemeldet und weggelassen; keine Handschrift, keine Akkorde, keine Mehrstimmigkeit. Die groesste Schwaeche: es gibt kein unabhaengiges OMR-Werkzeug auf dieser Maschine - die Rundlauf-Tests sind Selbstkonsistenz (auf verschlechterten Bildern und in zwei Massstaeben), und an einem echten Scan stimmen zwar die 6 Notenzeilen, nicht aber Taktzahl und Tonhoehen	XL

Ehrlich vorweg gesagt: MUS3 (die feinen Stichregeln) war der eigentliche Brocken und ist in M1426 abgeschlossen: Balkenraster (Sit/Straddle/Hang), melodische Konturneigung, Kollisionsausweichen fuer Bindeboegen, Fermaten und Beischriften sowie elastische Spaltenbreitenkompensation fuer gestaffelte Vorzeichen und Text.

Zur Verifikation: fuer eingelesenes MusicXML ist music21 (Python) ein unabhaengiger Leser, und

MuseScore/LilyPond koennen dasselbe Stueck zum Vergleich setzen. Nach der Projektregel geschieht

das einmalig beim Bauen; im Repo bleibt die eingebackene Erwartung, kein Fremdwerkzeug-Aufruf.

MUS13 · Der Noten-Editor im Einzelnen — 9 von 14

Nutzerwunsch (woertlich): *„ich moechte auch den noten editor so das er freistehend sein kann wie der editor und auch sonst alles naeher am editor so das man auch per drag drop oder per selektion und an der gewuenschten position die noten oder sonstige sachen plazieren kann und auch noten verbinden mit punkt leerem kreis noetenschluessel und all dem anderen was noetig ist. ich moechte auch ein klavier das unten ein und ausgeblendet werden kann mit dem man die noten selber eingeben kann und dazu die maus oder die tastatur verwenden kann mit einer tastatur mapping funktion in den settings."*

#	Scheibe	Stand
MUS13	Noten mit der Maus setzen: klicken an der gewuenschten Stelle, Tonhoehe aus der Zeile, Zeitstelle aus der Waagerechten	M1421 - MusicHitTest kehrt die Stech-Geometrie um (Bildpunkt -> Stufe ueber den Schluessel, X -> Takt und Zeitstelle), MusicEingabezeile zeichnet mit derselben Geometrie. Zwei ehrlich benannte Grenzen: getroffen wird das gleichmaessige Eingaberaster, nicht das gesetzte Bild (aus dem liesse sich nicht zurueckrechnen, welcher Bildpunkt zu welchem Eintrag gehoert); und der Klick bestimmt die Stelle in der Folge, nicht eine beliebige Zeitstelle - eine Luecke davor wird nicht mit Pausen gefuellt
MUS13	Paletten zum Anklicken: Notenwerte (ganz/halb/viertel… als leerer und gefuellter Kopf), Punktierung, Pausen, Notenschluessel, Vorzeichen, Taktangabe	Notenwerte, Punktierung, Pausen und Vorzeichen sind bedienbar. Offen: der Schluessel kommt heute nur ueber das Instrument der Spur, die Taktangabe nur ueber die Notenblatt-Definition - beides nicht als Palette
MUS13	Noten verbinden: Haltebogen, Bindebogen, Balkengruppen von Hand setzen und loesen	- Haltebogen und Balken sind im Stich gebaut (MUS3/M1404), das Setzen von Hand fehlt
MUS13	Tastatur-Zuordnung in den Einstellungen — frei belegbar und gespeichert	M1421 - der Rechenkern steht (MusicKeymap: frei belegbar, als Text speicher- und ladbar, kaputte Zeilen werden gemeldet und uebersprungen, eine unbrauchbare Eingabe faellt auf die Vorgabe zurueck und auch das wird gemeldet). Offen: die Bedienoberflaeche dafuer - der Editor nutzt heute die Vorgabe-Belegung
MUS13	Dynamik, Artikulation, Gesangstext und Akkordsymbole im Editor bedienen	- die Voraussetzung ist mit M1421 (MUS5/MUS6) gefallen: der Notensatz setzt jetzt alles davon. Was fehlt, ist allein die Bedienung im Editor

Die vier erledigten Unterpunkte (13.6, 13.8, 13.11, 13.12) stehen in roadmap-erledigt-v4.md.

Ehrlich zur Reihenfolge: MUS13.7 ist die Voraussetzung fuer 13.8–13.12 — solange aus einem Mausklick keine Tonhoehe und Zeitstelle wird, laesst sich nichts setzen, ziehen oder ueber eine

Klaviatur eingeben. Und 13.14 kann nicht vor MUS5/MUS6 fertig werden: was der Notensatz nicht zeichnet, kann der Editor nicht anbieten.

v4-PRN · Virtueller PDF-Drucker (Nutzerwunsch)

Auf jedem Geraet „Drucken" waehlen, das Ergebnis entsteht als PDF im Pruefstand und steht dort — mit Warteschlange — wieder zur Verfuegung. Bewusst KEIN Windows-Druckertreiber: der braeuchte ein signiertes Treiberpaket (WHQL/Code-Signing), einen Port-Monitor und liefe nur unter Windows. Der offene Weg ist ein IPP-Drucker (Internet Printing Protocol): Windows, macOS, iOS, Android und Linux drucken ohne jeden Treiber darauf, und moderne Clients senden den Job bereits als PDF.

#	Drucker-Funktion	Stand bei dir	Aufw
PRN3 7/7	Bekanntmachung im Netz (mDNS/Bonjour _ipp._tcp + AirPrint-Attribute) — der Drucker taucht von selbst in der Geraeteliste auf. Untertabelle unten: PRN3.1–3.7	- Unter Linux GEMESSEN (Debian-Container, .NET 9): auf Port 5353 sitzt dort meist schon ein Responder (avahi/mDNSResponder), doch SocketOptionName.ReuseAddress bildet auf Unix so ab, dass sich unser Socket dazubindet - belegt neben einem Halter mit SO_REUSEADDR (so bindet avahi) und auch benutzeruebergreifend (Halter als root, wir als anderer Benutzer); der Multicast-Beitritt gelingt ebenfalls. Gegenbeleg: haelt jemand den Port ganz ohne Reuse-Option, scheitert das Binden mit AddressAlreadyInUse - dann faengt der Dienst die Ausnahme und laeuft ohne Bekanntmachung weiter (der Pruefstand scheitert nicht). macOS bleibt ungeprueft (keine Maschine dafuer).	M

PRN3 · Der Drucker findet sich von selbst — 7 von 7

Nutzerwunsch (woertlich): *„beim spooler moechte ich auch wenn er im docker vorhanden ist das er als drucker im gleichen netzwerk segment erkannt wird und zb durch windows, mac, und linux automatisch den treiber holt und er sich installieren kann."*

#	Scheibe	Stand
PRN3.	End-zu-End-Abnahme je System (Windows, macOS, Linux, iOS, Android)	M1409 - gegen ein echtes CUPS abgenommen (docker/docker-compose.cups-abnahme.yml, scripts/cups-abnahme-starten.sh, Rueckgabewert = Ergebnis): ipptool meldet status-code = successful-ok, lpadmin -m everywhere richtet treiberlos ein, und zwei lp-Auftraege kamen als zwei PDFs an. Die Artefakte liegen als Fixtures im Repo - im Testlauf braucht es weder Docker noch CUPS. Ehrlich offen: Windows, macOS, iOS und Android sind ungeprueft - dafuer fehlt ein zweites Geraet im Netz. Dabei fiel eine Extraktor-Luecke auf: CUPS bettet Type0/Identity-H ohne /ToUnicode ein, PageTextExtractor liefert dann keinen Klartext (der Test gewinnt ihn ueber die cmap der eingebetteten Schrift zurueck)

Was „automatisch den Treiber holen“ wirklich heisst — die Unterscheidung entscheidet ueber den Aufwand: bei IPP Everywhere / AirPrint / Mopria gibt es keinen Treiber. Das Betriebssystem fragt den Drucker nach seinen Faehigkeiten und baut die Warteschlange daraus — genau deshalb ist PRN3.4 der eigentliche Punkt, nicht ein Download. Ein echtes Treiberpaket, das Windows aus dem Netz zieht und installiert, waere etwas anderes: seit Windows 10 ist das eine Print Support App aus dem Microsoft Store (v4-Treibermodell), also Store-Veroeffentlichung samt Signierung — nicht baubar ohne Store-Konto, und fuer einen virtuellen Drucker im eigenen Netz auch nicht noetig. Diese Grenze steht hier, damit sie nicht spaeter als Ueberraschung auftaucht.

v4-LR · Lightroom (Foto-Entwicklung und Katalog) — Nutzerwunsch

Die Bausteine dafuer sind zu grossen Teilen da: eigene Bild-Codecs (IMG), ICC-Farbmanagement mit echter Umrechnung (Icc), Resampler mit vier Verfahren (M1155), GPU-Kernel fuer Bildpunkt-Arbeit (Accel, M1023/M1031), Kontaktbogen (PhotoSheetComposer) und der PDF-/Druckweg. Was fehlt, ist die Foto-Entwicklung selbst und der Katalog.

#	Lightroom-Funktion	Stand bei dir	Aufw
LR1	RAW entwickeln: Demosaicing (Bayer/X-Trans), Weissabgleich als Temperatur/Tint, Kameraprofile	die Grundentwicklung steht mit M1309 (bilinear + Aufnahme-Weissabgleich); es fehlen X-Trans, Temperatur/Tint als Regler und echte Kameraprofile	XL
LR4	Details: Schaerfen (Betrag/Radius/Details/Maskieren) und Rauschreduzierung (Luminanz + Farbe)	Photo/Sharpening (M1404) - Unscharf-Maskieren mit Betrag, Radius, Details und Maskierung; die Kantenmaske war fehlnormiert (Nenner 510 statt 255, „Maskieren = 100" schaerfte heimlich mit ~70 %). Belegt ueber den entscheidenden Gegenbeleg: auf einer einfarbigen Flaeche aendert sich bei Betrag 300 kein einziger Bildpunkt - daran faellt eine „Schaerfung" auf, die in Wahrheit nur den Kontrast anhebt. Offen: die Rauschreduzierung ganz; Box-Filter statt Gauss, keine Halo-Begrenzung	L
LR5	Objektivkorrektur: Verzeichnung, chromatische Aberration, Vignettierung — rechnerisch und ueber Profile	fehlt	L
LR7	Lokale Anpassungen: Verlaufs- und Radialfilter, Pinsel, Bereichsreparatur — mit Masken	fehlt	XL

#	Lightroom-Funktion	Stand bei dir	Aufw
LR9	Nicht-destruktiv: die Bearbeitung ist ein Rezept, das Original bleibt unberuehrt; Sicherung als XMP-Sidecar	Photo/DevelopRecipe (M1404) - das Rezept mit seiner festen Reihenfolge (Weissabgleich -> Belichtung -> Kontrast -> die vier Tonwertbereiche -> Klarheit -> Dunst -> Dynamik -> Saettigung -> Kurven -> Mischer); es rechnet nichts Neues, sein Inhalt IST die Reihenfolge, und drei Tests belegen, dass sie nicht vertauschbar ist. Ein leeres Rezept gibt das Bild selbst zurueck. Offen: die Sicherung als XMP-Sidecar (setzt IMG15 voraus); das Rezept deckt nur die tonwert- und farbbezogenen Schritte ab - Geometrie, Schaerfen und die oertlichen Anpassungen laufen davor bzw. danach	M
LR10	Vorgaben (Presets) + Stapel-Synchronisierung ueber viele Bilder	fehlt	M
LR11	Katalog: Ordner/Sammlungen, Bewertung/Markierungen/Stichwoerter, Suche und Filter	fehlt (setzt IMG14/IMG15 voraus)	L
LR12	Export: Format/Groesse/Qualitaet, Ausgabeschaerfung, Wasserzeichen, Auswahl der mitgegebenen Metadaten	die Wandlung gibt es (convert_image, M1303), der Rest fehlt	M
LR13	Kontaktbogen / Web-Galerie / Fotobuch	Kontaktbogen da (PhotoSheetComposer), Galerie/Buch fehlen	M
LR14	GPU-Beschleunigung der Entwicklungs-Kette (die Kernel-Infrastruktur steht in Accel)	fehlt	L

Ehrliche Einordnung: LR1 und LR7 sind die beiden schweren Brocken; LR2/LR3/LR8 sind vergleichsweise geradlinige Bildpunkt-Arbeit auf dem vorhandenen Farb-Unterbau. Ein sinnvoller erster Schnitt waere LR2 + LR3 + LR8 + LR9 — damit gibt es eine echte, nicht-destruktive Entwicklung fuer JPEG/TIFF/PNG, noch ohne RAW.

v4-EXT · Extern/Toolchain-gebunden (kein fehlender Code)

Aus v3 hierher verschoben — kein fehlender Code, sondern extern/toolchain-/urteilsgebunden.

Punkt	Stand	Was noch fehlt / warum
.wasm-Artefakt (browser-aufrufbar, kpdf_*-Exporte)	zurueckgestellt	JS-Loader + Drift-Waechter (M975) stehen; braucht NativeAOT-LLVM aus dotnet/runtime-Quelle + emscripten-SDK. Auf Nutzerwunsch aufgeschoben.
Native-Shim signieren (Authenticode/codesign)	CI-Schritt	Rezept dist/build-native-shim.sh (M1007) steht; es fehlt nur das Code-Signing-Zertifikat als CI-Secret.
Docker-Hub Untagged-Speicher automatisch freigeben	zurueckgestellt	Tag-Loeschung ist automatisiert (Cleanup Phase A, taeglich). Untagged-Digests belegen ~22 GB; belegt (3 gruene Cleanup-Laeufe lassen sie liegen): Docker Hub hat KEINE oeffentliche REST-API dafuer (Image-Management-UI = session-gebundener Loader). Einzige Automatik = ganzes Repo vor jedem Push loeschen — macht das Repo aber PRIVAT + verliert Beschreibung/Pull-Zaehler. Auf Nutzerwunsch aufgeschoben; bei Bedarf manuell ueber die Web-UI (Image Management) oder Docker-Hubs 6-Monats-GC.