Roadmap — erledigt in v4

Farb-Legende (Status):

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

Teil der Roadmap — dort steht, was noch offen ist.

v4-ROB · Erschoepfung durch begrenzte Eingabe

Punkt	Was	Stand	Aufw.
ROB5	Der Waechter, der die Klasse dauerhaft faengt statt sie einzeln zu suchen: ein Fuzz-Lauf ueber alle Leseformate mit Zeitbudget je Fall (abgeschnitten an jeder Byte-Grenze, verfaelscht mit festem Startwert). Ein Fall, der sein Budget reisst, ist ein Befund — genau das haette M1451 sofort gezeigt, statt dass es zufaellig bei einer Frage nach der Testdauer auffiel. Der wertvollste Punkt der Gruppe, und der einzige, der auch kuenftige Leser abdeckt	M1453	M
ROB6	Die Schrift-Leser im Erschoepfungs-Waechter	M1506 - erst gemessen: eine Sonde hat zwoelf Schriftproben (TrueType einfach/zusammengesetzt, CFF, CID-CFF, CFF2, gvar, WOFF, WOFF2, COLR v0/v1/v1 mit Variationen, ein Ausschnitt aus Noto Sans Bold) mit zusammen 49 321 abgeschnittenen und verfaelschten Faellen einzeln gestoppt: kein Fall ueber 1 s (Spitze 907 ms bei CFF2), also keine Erschoepfung - aber VariableFont.Instantiate liess an vier Stellen (Punktnummern, gepackte Deltas, Tupel-Kopf, Glyphenumriss) eine IndexOutOfRangeException durch, jetzt InvalidDataException. Danach verdrahtet: ErschoepfungsWaechterTests.Schriftproben liest jede Probe auf EINEM Weg (Behaelter entpacken, Struktur pruefen, parsen, jeden Glyphen lesen, Variationen instanziieren, COLR v0/v1, Einbettung). Ein Waechter, der nichts faengt, belegt nichts: mit abgeklemmtem COLRv1-Knotendeckel blieb der Waechter zuerst gruen - die Bomben-Probe faechert nur auf 520 200 Blaetter auf; erst die volle Verfaelschung der Bombe fand den Zyklus (Zaehlfeld an Versatz 2863, 5 min 41 s). Er steht jetzt als eigene Probe COLRv1-Zyklus im Waechter (wie die ROB2-Befunde), und genau er wird ohne Deckel rot	M

MG · Gemeldete Maengel an bereits Gebautem

Punkt	Was gemeldet ist	Was zu tun ist	Aufw.	Stand	
MG1 · Einband-Gestaltung	Der Cover-Weg (Bibliothek M1159, REST + Editor) „funktioniert nicht wie es sollte" — die erzeugte Vorlage entspricht nicht der Erwartung	Am konkreten Fall nachmessen: Bogenmasse, Ruecken, Falzlinien, Klappen gegen ein von Hand gerechnetes Soll; die Abweichung benennen, dann beheben. Erledigt (M1319). Die Geometrie war korrekt (gegen Handrechnung nachgemessen); die Falzlinien sind eine /Separation-Sonderfarbe, und der Rasterizer las den Tint-Wert als Helligkeit — Vollton wurde reines Weiss. Die Anzeigeliste loest den Farbraum jetzt auf (IsSeparationSpace + SeparationTint)	M		
MG2 · Montageflaeche im Editor	Die Montageflaeche (M1228 Kern, M1229 Oberflaeche) „funktioniert nicht wie es sollte"	Nachstellen: Objekt auf die Flaeche ziehen, speichern, neu laden, PDF erzeugen. Erledigt (M1319). Modell und .kdoc-Rundlauf waren korrekt; die Anzeige-Felder wurden nur beim Oeffnen des Eigenschaften-Dialogs nachgezogen, nicht bei jedem Dokumentwechsel (Oeffnen/Neu/Einband/Importieren/Undo/Redo)	M		
MG3 · Editor: Seitenwechsel sehr langsam	Beim Wechseln der Seite war der Editor „extrem langsam"	Erledigt (M1317). Gemessen: 667 ms je Wechsel — der Hintergrund hielt genau EINE Seite und rechnete bei 150 dpi neu, auch beim Zurueckblaettern. Jetzt ein Speicher je Seite (RenderCache, testbar) und 96 dpi	M		
MG5 · Import: Text kommt als Kaestchen an	Eine echte Rechnung (KPT) zeigte im Editor statt Text nur eine Reihe kleiner dunkler Kaestchen	Erledigt (M1373). Am Dokument gemessen: der Erzeuger setzt /F2 1 Tf und traegt die Groesse in die Textmatrix (10 0 0 10 x y Tm) — der Import nahm nur die Tf-Zahl und schrieb fontSize: 1. Weil die Absatz-Buendelung (M1314) ihre Grenzen in Vielfachen der Groesse misst, zerfiel damit jede Zeile in Ein-Zeichen-Objekte (799 Objekte auf Seite 1). Neu TextRun.EffectiveFontSize = Laenge des transformierten y-Einheitsvektors der Renderingmatrix (gilt auch gedreht/geschert), im Import ueber VisibleFontSize genutzt. Nachher: 43 Objekte, Groessen 6–14 pt	S		

Punkt	Was gemeldet ist	Was zu tun ist	Aufw.	Stand	
MG6 · Voller Testlauf extrem langsam	„wie so benoetigt der volle lauf so extrem lange?" — 18 min 49 s fuer 15'474 Tests	Erledigt (M1451). Mit TRX-Protokoll vermessen statt geraten: ein Test kostete 1145 s = 97 % der Wanduhr (xunit parallelisiert je Testklasse ueber 32 Kerne, die Dauer ist also die laengste Klasse, nicht die Summe von 4398 s). Dahinter steckte kein Testmangel, sondern eine Luecke im Robustheitsprinzip: MqDecoder fuellt am Datenende und an einer Marke normgerecht mit 1-Bits auf (T.88 §E.3.4), hoerte damit aber nie auf - und die Textregion nimmt eine Zahl aus dem Strom als Schleifengrenze (SBNUMINSTANCES, bis 16,7 Mio.). Gemessen: 115 Byte Eingabe erzeugten 16 min 28 s Rechenzeit, ohne dass eine Ausnahme fiel. Damit ist die Zusicherung „aus Load dringt keine Ausnahme" als halbe Zusicherung entlarvt - „kommt in endlicher Zeit zurueck" gehoert dazu. Neu MqDecoder.LiestNurNochErfundeneDaten (zaehlt die BYTEIN mit stillstehendem Lesezeiger); die vier Stellen mit einer Stromzahl als Grenze brechen ab. Voller Lauf 18:49 -> 4:16, die 118 JBIG2-Tests 1145 s -> 116 ms	M		
MG7 · Noten-Editor nutzt die gebuendelte Notenschrift nicht	Der Editor meldet zugleich „Keine SMuFL-Notenschrift uebergeben" und „Die uebergebene Notenschrift liess sich nicht lesen" — obwohl Bravura seit M1426 ab Werk gebuendelt ist (MUS2)	Ursache im Code nachgesehen, nicht geraten: MusicEditor.razor setzt MusicFont = _schrift is null ? null : _schrift. Der Ausdruck hat den Typ byte[], und ein null-byte[] wird beim Uebergang nach ReadOnlyMemory<byte>? zu einem leeren, nicht zu null — dieselbe Ternaer-Falle wie beim poster-Argument in M1348. MusicPdf liest „leer" als ausdrueckliches keine Schrift und laesst die Vorgabe Bravura weg; weil der Wert nicht null ist, kommt zusaetzlich die Lese-Meldung. Folge: ohne selbst geladene Schrift fehlen im Editor Schluessel, Taktangabe, Pausen und Vorzeichen. Zu tun: nur bei geladener Schrift uebergeben, sonst die Eigenschaft gar nicht setzen; Test durch den echten Dienst (Schluessel wird gezeichnet, keine der zwei Meldungen). Dazu den Seitentext nachziehen, der noch sagt, ohne mitgelieferte Schrift fehlten diese Zeichen	S	S	Erledigt (M1473).
MG8 · Noten-Editor: veralteter Hinweistext	„Was dieser Editor nicht kann" sagt, die zweite (und dritte) Zeile von Klavier/Orgel setze der Notenstich noch nicht	Seit M1457 (MUS7) setzt der Notenstich mehrere Notenzeilen je Stimme. Offen ist allein das Bearbeiten auf der zweiten Zeile im Editor (MUS13.15) — der Text muss das so sagen, statt den Satz als Grenze zu nennen	S	S	Erledigt (M1473).

v4-KDX · .kdoc als Vollabbild — die ersten zwei Bauwellen (M1438–M1450)

Ziel der Gruppe: Import -> .kdoc -> Export ins selbe Format verliert nichts. Befund und Entwurf in docs/superpowers/specs/2026-09-19-kdoc-vollabbild.md. Was noch offen ist, steht in roadmap.md.

#	Punkt	Aufw.	Stand
KDX1	Nie gespeicherte Laufzeitfelder	L	M1438. Das .kdoc sichert jetzt Objekt-, Tabellen- und Zellformate, Textvariablen, bedingten Text, Zeilennummerierung, Beschnittform und Ueberlauf, Eckenradien, Min/Max-Masse, Umriss und Transformation samt Ursprung, ShapeFrame-Konturen/Mehrfachlinien/Kachelmuster/Bildfuellung/Ueberdrucken/Pfadverlauf, den TextStyle-Rest (Zeichenabstand, OpenType-Merkmale, Textschatten, Absatzlinien, Grundlinienraster, optischer Randausgleich), Lesezeichen/Querverweis/Float am Rahmen, XMP, PDF/A-, PDF/X- und PDF/UA-Kennung, die volle Ausgabeabsicht samt eingebettetem ICC-Profil, Werkzeuglinien - dazu die drei Faelle, an denen das Speichern bisher den ganzen Vorgang scheitern liess (eingebettete Schrift im Stil und am Feld, Sonderfarbe am Formularfeld, TikzFrame/FloatClearMarker). Formatversion bleibt 1, ohne die neuen Angaben byte-identisch. Zwei Auftragspunkte lagen beim Messen anders: der weiche Umbruch war bereits in Ordnung (vier Tests als Beleg), und die Spiegelung an allen Rahmen ist bewusst nicht gebaut - sie waere ein gespeichertes Feld, das kein Rahmen zeichnet (jetzt KDX12). Grenzen: Sonderfarbraum an einer Format-Linie wird gemeldet statt gespeichert; Encrypt/AttachFile/ConfigureBuilder legen Rueckrufe ab und sind grundsaetzlich nicht speicherbar
KDX2	Fliesstext-Modell	XL	M1442-M1445 (vier Pakete). Ueberschriften bis Ebene 9 (die Klemme auf 1..3 war der Befund aus M1441; StructureTag kannte H1-H6, H7-H9 fehlten - sie stehen erst in ISO 32000-2, ein PDF/A-1a/2a-Pruefer kann sie bemaengeln), Umbruchsteuerung am Absatz (zusammenhalten, Umbruch davor/danach, Witwenschutz je Absatz statt nur dokumentweit) und Zeilenhoehen genau/mindestens/Vielfaches. Je Lauf tragen Link, Sprungziel, Sprache (als /Span <</Lang ...>> ohne MCID, damit ein Absatz aus fuenf Laeufen EIN /P bleibt), Hervorhebung und Querverweis - die Grenze, die FlowDocument.CrossReference selbst benannte („kein Inline-Stueck mitten in einem Satz“), ist damit weg; der Querverweis wirkt beim Setzen, nicht beim Zeichnen, sonst liefen Messen und Zeichnen auseinander. Noten, Kommentare, Aenderungsverfolgung als eigener .kdoc-Teil annotations.json: die Note kennt ihren Anker, der Kommentar seine Antwortkette und seinen Erledigt-Zustand, und eine einzelne Aenderung laesst sich annehmen oder ablehnen - genau die Grenze, die InCopy/TrackedText (M1088) benennt. Der Anker geht ueber den Namen des Rahmens plus Zeichenversatz, nicht ueber Koordinaten (dieselbe Entscheidung wie MasterPageName). Abschnitte mit eigenem Seitenformat, Raendern, Spalten, Kopf-/Fusszeile (erste/gerade/ungerade getrennt) und Neubeginn der Seitennummerierung - ueber das vorhandene FlowRegion, das sein eigenes PageSize schon hatte, also ohne zweiten Paginierungsweg; mehrstufige Listen mit Marker und Einzug je Ebene und ausgezeichneten Laeufen; ein Feld-Verzeichnis (Art, Befehl, letzter Wert) am vorhandenen TextVariables statt eines zweiten Mechanismus; sechs fehlende Metadaten. Gemessen statt gebaut: KeepWithNext war da, vier der zehn Metadaten waren da, TextVariables lieferte den Anker im Text - nichts davon wurde neu gebaut. Formatversion bleibt 1, ohne die neuen Angaben byte-identisch. Offen und als eigene Zeilen benannt: die Liste im Fliesstext (KDX15) und die Anbindung der Importer (KDX16)

#	Punkt	Aufw.	Stand
KDX7a	InDesign nativ lesen und schreiben	XL	KDX7. .idml geht in BEIDE Richtungen direkt, nicht mehr ueber ein zwischengerendertes PDF. Der neue IdmlNativeReader liest Spreads mit Seiten und Rahmen samt Geometrie, Stories mit Absaetzen/Laeufen/Tabellen/Ankern, Formate samt BasedOn-Kette (mit Ringschluss-Schutz), Farbfelder, Musterseiten, Ebenen und Bedingungen; IdmlToKdoc bildet jede Seite eines Spreads auf eine Layout-Seite und jeden Rahmen auf ein anfassbares Objekt ab, KdocToIdml schreibt den Weg zurueck. Der Zuschnitt ist ein Zwischenweg und begruendet: Seiten aus Rahmen wie bei Word/Visio/PowerPoint, dazu ein schmaler Bestand (LayoutDocument.IdmlResources, .kdoc-Schema §8.29) fuer Farbfelder, Formate, Musterseiten und Bedingungen - die stehen nirgends auf der Seite und gingen ueber die Geometrie zwangslaeufig verloren. Story, Verkettung, Spaltenzahl und Bedingungen haengen am Objekt selbst. Tabellen werden zu TableFrame (Spaltenbreiten, Zellinhalte, Kopfzeilen, verbundene Zellen), bedingter Text WIRKT (ein Lauf unter einer unsichtbar geschalteten Bedingung wird nicht gesetzt; eine unbekannte Bedingung gilt als sichtbar), verankerte Objekte kommen an. Es gibt jetzt genau EINEN IDML-Leser und EINEN IDML-Schreiber - IdmlReader, IdmlToPdf und IdmlWriter sind Fassaden darauf. Fremdorakel SimpleIDML (BSD-3-Clause; die fruehere Testdoku behauptete, es gebe keines - die siebte falsche Behauptung dieser Art): drei echte, von InDesign geschriebene Pakete liegen als Fixture bei, und SimpleIDML liest den vollen Rundlauf. Vier Fehler, die nur echte Dateien oder ein fremder Leser zeigen konnten: die CharacterStyleRange steckt in <XMLElement>, sobald das Dokument XML-Tagging traegt (der alte Leser fand deshalb in einer echten Datei KEINEN Buchstaben); das geschriebene Paket trug keine Resources/Fonts.xml; eine Form verwies auf ein Farbfeld, das im Paket gar nicht angelegt war; und der Import wies Objekte einer Ebene zu, ohne sie anzulegen - das .kdoc liess sich dadurch GAR NICHT speichern. Dazu ein Rahmen mit Hoehe 0, der als groessenlos verworfen wurde (dieselbe Falle wie bei Word, M1432)
KDX3	Fliesstext-Import direkt	L	M1441. RTF, ODT (neuer OdtReader), FB2, EPUB, HTML, Markdown und TXT gehen ueber ihren eigenen Leser ins .kdoc statt ueber ein zwischengerendertes PDF; der DocxReader nimmt jetzt Formatvorlagen (mit basedOn/docDefaults), gridSpan/vMerge, Tabellen- und Zellrahmen je Kante, Absatzformat und core.xml mit; RTF liest Schrifttabelle, Formatvorlagen, Einzuege/Abstaende, Durchstreichen, Hoch-/Tiefstellung, Kapitaelchen, \info und \pict; EPUB und FB2 liefern ihre Metadaten, EPUB zusaetzlich die Bilder aus dem ZIP; TXT erkennt seine Kodierung statt fest UTF-8. Drei Altfehler dabei: <title> zaehlt zu den unsichtbaren Elementen, der HTML-Metadatenweg lieferte immer ""; ReadElementContentAsString verschiebt den Leser bereits, dadurch wurde jedes zweite dc:-Element uebersprungen; im RTF stehen die Steuerworte eines Stylesheet-Eintrags vor dem Namen. Zwei behauptete Grenzen waren falsch (RTF \sa/\sl, ODT fo:margin-bottom/fo:line-height - beide haben laengst ein Feld) und sind berichtigt

#	Punkt	Aufw.	Stand
KDX4	Fliesstext-Export direkt	L	M1446-M1450 (drei Pakete plus das Einhaengen). .docx/.odt (M1446), .rtf/.epub/.fb2 (M1447) und .txt/.html/.md (M1448) gehen unmittelbar aus dem Arbeitsdokument hinaus; HTML und Markdown sind neu ueberhaupt Zielformate. Wie teuer der PDF-Umweg war, ist gemessen, nicht behauptet: eine Seite aus einem Absatz und einer 2×2-Tabelle kam als EINE Tabelle mit vier Spalten zurueck - der Absatz war mit hineingezogen, die Kopfzeile hatte es nie gegeben. Dafuer fehlte die Umkehrung zu ToUnrenderedLayout: neu Layout/LayoutContentExport (Objektmodell -> Export-Sicht) und Layout/LayoutFlowWalk (Objektmodell -> Lesereihenfolge). Jeder Schreiber hat jetzt zwei Eingaenge und EINEN Rumpf - ein zweiter Schreibweg je Einstieg waere eine zweite Wahrheit. Beim Zusammenfuehren drei Entscheidungen, die kein Werkzeug treffen kann: die Pakete A und C hatten dieselben Laufmerkmale unabhaengig gebaut, teils unter verschiedenen Namen (aufgeloest als Vereinigung, Name vereinheitlicht auf RunLinkUrl - LinkUrl steht an FlowContentItem bereits fuer den ganzen Block); jede Seite trug etwas, das der anderen fehlte (Sprungziel am Lauf nur beim Fassaden-Weg, Durchstreichen/Grundlinienversatz/Schriftname nur beim direkten - sonst haenge es am Einstieg, ob ein durchgestrichener Lauf im .docx durchgestrichen ankommt); und LayoutFlowWalk zaehlte einspaltige und mehrspaltige Abschnitte getrennt auf, sodass ein mehrspaltiger Abschnitt ZWISCHEN zwei einspaltigen ans Ende rutschte - das Modell fuehrt dafuer seit M1432 FlowContainersInOrder (mit rotem Test belegt, nicht vermutet). Zwei Fehler fand erst das Gegenlesen: die Fussnotenmarke sass am Anfang des Laufs statt an ihrer Zeichenstelle, und eine Ueberschrift bekam ihre Praesentation als feste Punktgroesse hineingeschrieben - in einem reflowbaren Buch uebergeht das die Leseeinstellung. Ein eigener Fehler: ein benanntes Absatzformat wurde referenziert, aber nie definiert; Word und LibreOffice fallen dann still auf den Basisstil zurueck. Orakel: Word 16 (COM), LibreOffice 26.2, python-docx, odfpy, ebooklib, html.parser/lxml.html, python-markdown - sie fanden u. a., dass ODT gar keinen Zellverbund schrieb. Fuer .txt gibt es ausdruecklich keines (reiner Text hat keine Grammatik zum Gegenlesen), und das steht so in der Testdoku statt eines vorgetaeuschten Fremdbelegs. Grenzen: Bilder/Formen/Diagramme traegt das Fliessmodell nicht; ODT schreibt die Aenderungsverfolgung als Endfassung; HTML-Ebenen 7-9 sind ARIA-korrekt, ueberleben aber den eigenen Rundlauf nicht; Kommentare, Aenderungsverfolgung und Abschnitte erreichen .html/.md/.txt noch nicht (KDX18)
KDX8	Visio-Rest	L	M1449. Shape-Daten (Property), Hyperlinks, Verbindungspunkte als Sektion, Kommentare (Annotation/Reviewer), die Feldidentitaet (ein <fld> kam bisher als blosser Text an), die Formvorlagen-Referenz (der zellweise Master-Verschmelzer loest die Herkunft auf, sie wird deshalb vor dem Verschmelzen festgehalten) und QuickStyle - alles, was ein Visio-Dokument fachlich ausmacht und genau deshalb durchfiel, weil es nichts zeichnet. Der Schreiber legt einen echten Vorlagen-Teil an, damit die Vorlage den Rundlauf ueberlebt; Spiegelung und 3D-Tiefe werden jetzt gesichert statt gemeldet. Vier der zehn Roadmap-Punkte standen bereits (Themenfarben und Ebenen M1436, Spiegelung im PDF-Weg M1437, mehrere Geometrien M1434) - am Code geprueft statt neu gebaut. Dabei die Faktor-72-Falle aus M1429 erkannt und umschifft (Lese-Kommentar in Zoll, Schreibmodell in Punkt - nichts stuerzt ab, die Zeichnung ist nur winzig): eigener Typ, Falle in der Klassendoku benannt. Kein Orakel, und das steht so da: Property, Hyperlink, Connection, Field und Annotation sind reine Datensektionen, die kein Betrachter zeichnet - belegt wird ueber den Rundlauf durch den anderen Codepfad. Grenzen: User/Actions/Control/Scratch/SmartTagDef werden gemeldet, nicht getragen (Visio-Laufzeitlogik, kein Inhalt); Deckel 4096 je Liste mit Meldung statt stillem Kappen; von den Anmerkungen kehrt nur Note zurueck

#	Punkt	Aufw.	Stand
KDX9	PDF vollstaendig	XL	M1439. Der Import nimmt Gliederung mit Zielseite, Info-Metadaten, /Lang, Trapped, Seitenbeschriftungen, Anzeigeeinstellungen, OCG-Ebenen samt Bindung jedes Objekts ueber seine BDC/EMC-Klammer, /Rotate, Trim-/BleedBox, echte Kurven als ShapeKind.Path, Strichmuster, Linienenden, Deckkraft, Mischmodus, achsiale und radiale Verlaeufe, Textfarbe, interne Links auf eigener Flaeche und die Formularfeld-Merkmale; die Malreihenfolge bleibt erhalten, und unsichtbarer OCR-Text (Rendermodus 3) wird erkannt und verworfen statt sichtbar gemacht. Im Export schreibt der neue PageOverlayWriter Annotationen, Schwaerzungsmarken und Signaturfelder - vorher fielen alle drei still weg. Ein Befund richtiggestellt: die 1-Kanal-Separation wurde als Grauwert gedeutet, dort ist die Zahl die Tintendeckung. Grenzen: Strichphase, Kachelmuster, Mesh-Schattierungen, sh ohne Pfad und Clip-Pfade; die Modellfelder, die dafuer fehlten, sind mit KDX13 geschlossen
KDX13	PDF-Modellfelder (Befunde M1439)	L	Etappe A (Antwort-Ketten/Status an Kommentaren ueber einen dokument-eigenen Namen, QuadPoints/Vertices als echte Zeilen-Vierecke, Signatur-Grund/Ort/Kontakt eines bereits signierten Feldes), Etappe B (Formularaktionen /A+/AA als FieldActionScript - gespeichert, NICHT ausgefuehrt, das bleibt dem opt-in Scripting-Add-on vorbehalten -, /AcroForm /CO, mehrere Widgets je Feld ueber /Kids) und Etappe C (diese, fuenf Teilpunkte). Alt-/ActualText: LayoutObject.StructureActualText (Gegenstueck zu StructureAltText), im .kdoc bisher GAR NICHT gespeichert - der im Auftrag benannte Verlust. PDF-Import korreliert /Figure-Struktur-Elemente ueber die MCID der /Figure <</MCID n>> BDC ... Do ... EMC-Klammer mit dem gemalten Bild (dieselbe Stapel-Mechanik wie TaggedContentExtractor, dessen ReadMcid/CurrentMcid dafuer geteilt statt verdoppelt werden). Rendermodus am Lauf: TextStyleLookup.RenderModeOf traegt jetzt den echten PDF-Rendermodus (0..7) statt ihn auf Fuellen zu vereinfachen; Modus 3/7 (die OCR-Textebene) bleibt bewusst weiterhin uebergangen. Ein Korrektheits-Fund dabei: AppearanceColors.ApplyTextFill setzte nur die Fuellfarbe - ein Konturmodus haette beim Zurueckschreiben mit der zufaellig aktiven Konturfarbe umrandet; PageCanvas.DrawWordForStyle setzt sie jetzt explizit. Clip-Pfad: neues ShapeFrame.ClipPathShape (die ClipShapeRenderer-Rechnung aus BoxFrame herausgezogen, damit beide dieselbe Beschnitt-Mathematik nutzen); TryBuildImportedClip liest die aktive Beschnitt-Region aus dem Grafikzustand zum Zeitpunkt des Malens. Kachelmuster + Mesh-Schattierung als Fuellung: TryBuildTilingPattern erkennt /PatternType 1 mit leerem /Resources (sonst waere der Zelleninhalt beim Zurueckschreiben kaputt) und ruft TilingPattern.FromContent unveraendert; neu Layout/MeshShadingFill + ShapeFrame.FillMeshShading fuer /ShadingType 4..7 - seiten-verankert wie das Kachelmuster (nur bei Seitenursprung (0,0) und identischer Matrix, drei Geraete-Farbraeume). OCG-Verschachtelung: neu PdfLayer.ParentName (namensbasiert wie LayerNames/MasterPageName), PdfDocumentBuilder.BuildLayerOrder baut daraus eine verschachtelte /D /Order-Struktur (Tabelle 100) mit Zyklen-/Unbekannt-Wache; /OCMD (mehrere OCGs mit boolscher Politik an einem Objekt) bleibt bewusst aussen vor. Ein echter Rasterizer-Fund beim Testen: PathRasterizer.Render(DisplayList, scale) liest KEINE Seiten-Ressourcen - jede Muster-/Schattierungsfuellung blieb damit ungemalt, auch am unveraenderten Original; erst Render(DisplayList, PdfDocument, scale) malt sie, und damit BEIDE Fuellungen nachweislich (M123, ueber Pixel: eine Kachel wiederholt sich wirklich, eine Mesh-Fuellung deckt genau ihre Dreiecke ab). ImageFrame/TextFrame bekommen weder Clip noch Muster-/Mesh-Fuellung (nur ShapeFrame - der Umfang war je Teilpunkt bereits gross genug)

#	Punkt	Aufw.	Stand
KDX10	SVG/EMF/WMF/ODG	M	M1440. SVG geht als ein Rahmen mit den Originalbytes direkt ins Objektmodell; EMF/WMF gingen zunaechst ueber MetafileToLayout (Zerlegung in einzeln editierbare Formen - die Originalbytes gingen dabei verloren, geschlossen von KDX14); neuer OdgReader; der ODG-Export nimmt OdgWriter.Write(LayoutDocument) direkt statt Render->PdfToOdg, PowerPoint ebenso, und .svg ist neu exportierbar (LayoutSvgWriter; bei genau einem SVG-Rahmen gehen dessen Rohbytes unveraendert hinaus). NeedsRenderedPath faellt fuer Fluss-Container und Musterseiten begruendet auf den alten Weg zurueck. Zwei Sabotage-Proben widerlegten die eigenen Tests: der gerenderte Weg liefert Rechteck und Ellipse sehr wohl als native Form - die Unterscheider sind jetzt die erhaltene Gruppe (ODG) und die Rahmenhoehe (PPTX). Grenzen: Verlaeufe, Fuellmuster, Effekte und Ebenen-Sichtbarkeit kommen aus ODG nicht mit
KDX14	MetafileFrame mit Originalbytes	S	M1455. Genau der KDX10-Rest: der direkte EMF/WMF-Importweg (VectorLayoutImport) baut jetzt - wie bei SVG - EINEN opaken MetafileFrame mit den Originalbytes, statt die Zeichnung in einzelne Formen zu zerlegen. Gezeichnet wird weiterhin ueber den vorhandenen Malweg (neu MetafileToPdf.PaintOnto, oeffentlich gemacht) - kein zweiter Zeichenweg. Der .kdoc-Rundlauf und der Export folgen dem SvgFrame-Muster: KdocObjectKind.Metafile traegt die Bytes als eigene Ressourcendatei (wie "svg"/"eps"/"pdfPage"), und DocumentExporter gibt sie unter .emf/.wmf nur dann unveraendert zurueck, wenn die Byte-Kennung (MetafileFrame.IsEmf) zur verlangten Endung passt - sonst der bisherige, gerenderte Weg. M123-Gegenbeleg, mit rotem Test belegt: dieselben Bytes ueber den AELTEREN, zerlegenden Weg (MetafileToLayout.Convert, bleibt oeffentlich aufrufbar fuer wer einzeln editierbare Formen will) kommen beim Export NICHT byte-genau zurueck. Grenzen: die Groesse der .kdoc-Datei waechst wie bei jeder anderen eingebetteten Ressource (kein gesonderter Deckel)

#	Punkt	Aufw.	Stand
KDX12	Spiegelung an allen Rahmen	M	Der gemeinsame Zeichen-Rahmen. Gemessen statt vermutet: 16 Zeichenmethoden in 15 Rahmenarten (Shape/Text/Image/ObjectGroup/List/Barcode/Chart/DropCap/Eps/Metafile/PdfPage/ReferenceXObject/ Svg/TextPath/TableFrame×2) wendeten LayoutObject.Rotation je einzeln ueber canvas.RotateAround(...) an - die Roadmap-Schaetzung „~15" stimmte. Neu PageCanvas.MirrorAround(flipH, flipV, cx, cy) (eine Matrix wie RotateAround, No-op ohne Achse) und LayoutObject.FlipH/FlipV + der gemeinsame Zeichen-Rahmen ApplyObjectTransform(canvas, rotation, cx, cy) - die EINE Stelle, die Spiegelung UND Drehung in eine Matrix uebersetzt. Alle 16 Aufrufstellen sind darauf umgestellt; ohne FlipH/FlipV byte-identisch (124 bestehende Rotation-Tests unveraendert gruen). Die Reihenfolge war beim ersten Anlauf FALSCH: PDFs cm-Verkettung wirkt von innen nach aussen (die zuletzt emittierte Matrix wirkt auf den Punkt zuerst) - die eigene Pixel-Probe fand es (FlipH+Rotation(90) landete an der falschen Ecke), behoben durch die Emissions-Reihenfolge (RotateAround VOR MirrorAround). DrawingMlShapeObject.FlipH/FlipV (die bestehende, geometrie-basierte Spiegelung fuer gelesene DrawingML-Formen) bleibt bewusst getrennt (new bool FlipH/FlipV, dieselbe Entscheidung wie RotationDeg neben Rotation - andere Semantik, anderer Draw-Pass). Im .kdoc gesichert (nur wenn gesetzt, Formatversion bleibt 1). Verifikation: 8 diskriminierende Pixel-Tests ueber PathRasterizer/DisplayListBuilder - FlipH allein spiegelt waagerecht und NICHT senkrecht (Gegenbeleg zu FlipV) und umgekehrt; FlipH+Rotation(90) ergibt eine ANDERE Lage als beide fuer sich; die verworfene Reihenfolge von Hand ueber PageCanvas reproduziert und als tatsaechlich andere Lage belegt; ImageFrame (asymmetrisches 2×1-Bild) und ObjectGroup (zwei Kind-Shapes) nutzen denselben Zeichen-Rahmen. Dabei ein Befund an ObjectGroup: sie SCHRUMPFT ihre FrameSize immer auf die Bounding-Box ihrer Kinder (eigene Layout-Regel, FrameSize = boundingBox;) - der Dreh-/Spiegelpunkt ist also deren Mitte, nicht eine von aussen gesetzte Groesse. Ehrlich benannte Grenze: der PDF-Import rekonstruiert FlipH/FlipV nicht aus einer gespiegelten CTM (Matrix-Dekomposition waere ein eigener, groesserer Baustein) - ein fremdes PDF mit gespiegeltem Inhalt kommt weiterhin mit bereits gespiegelten Koordinaten herein, nicht mit gesetztem FlipH; der PDF-Export ist davon nicht betroffen, da Draw() die Matrix ohnehin direkt in den Content-Stream schreibt (belegt ueber denselben Pixel-Test - Render -> Bytes -> PathRasterizer)

#	Punkt	Aufw.	Stand
KDX5	PowerPoint direkt in beide Richtungen	XL	M1469. PptxToKdoc/KdocToPptx statt des PDF-Umwegs - wie teuer der war, ist gemessen: eine Tabelle kam ueber das gerenderte PDF als loses Strich- und Textgemenge zurueck, nichts mehr Anfassbares. Gebaut in drei Scheiben: (1) die zwei Richtungen samt Originalteile-Speicher fuer .pptx (docProps/custom.xml ueberlebt, mit Rundlauf-Waechter - dieselbe Entscheidung wie bei .docx in KDX11); (2) Uebergang und Zeitsteuerung als Rohdaten (PptxSlideInfo.TransitionXml/TimingXml), Abschnitte, Hintergrund, Thema, Diagramme als ChartFrame, SmartArt ueber ppt/diagrams/drawingN.xml, Gruppen als ObjectGroup, Kommentare je Folie; (3) Gruppen bleiben beim Schreiben Gruppen (echtes <p:grpSp> mit <a:xfrm>), SmartArt gegen eine echte Fixture belegt. Zwei Funde, die nur ein Orakel liefern konnte. (a) PowerPoint verpackt <p:transition> in mc:AlternateContent - der direkte Kind-Blick des Lesers fand es nie, und weil Leser und Schreiber denselben Irrtum teilten, konnte kein Rundlauf es zeigen. (b) chOff muss gleich off sein: eine nach Spezifikation gueltige Fassung mit chOff=(0,0) loeste der vorhandene Leser bei zwei Verschachtelungsebenen falsch auf (Kind bei 100/100 statt 50/50) - der Schreiber folgt jetzt PowerPoints eigener Konvention. Die SmartArt-Fixture, in zwei Anlaeufen. Application.SmartArtLayouts.Count ist auf dieser Maschine 0 (per COM gemessen, von mir nachgeprueft) - AddSmartArt ist hier nicht aufrufbar. Der erste Anlauf baute die PPTX-Huelle deshalb von Hand: sie trug 9 Teile und WEDER [Content_Types].xml NOCH _rels/.rels, war also gar kein gueltiges OPC-Paket. Unser eigener Leser nahm sie an, weil er tolerant ist - der Test belegte damit nur, dass unser Leser unsere eigene Datei liest. Aufgefallen ist es, weil LibreOffice gar nicht erst geprueft worden war; es meldet "source file could not be loaded", waehrend es die zwei von PowerPoint erzeugten Fixturen (37 bzw. 39 Teile) als PDF exportiert. Die Huelle stammt jetzt aus PowerPoint (42 Teile); belegt ist sie in beide Richtungen: LibreOffice liefert ein PDF von 16275 Bytes gegen 7048 fuer dieselbe Huelle ohne die Zeichnung (die Grafik kommt also wirklich an, statt dass nur eine leere Folie durchlaeuft), und PowerPoint meldet genau ein msoSmartArt-Shape mit drei Knoten. Die fuenf Diagramm-Teile sind byte-gleich zur Word-COM-Fixture (M1431) geblieben. Beim Einsetzen ein dritter Fund: die Beziehungs-IDs sind nicht frei waehlbar - der Datenteil nennt seine Zeichnung selbst in <dsp:dataModelExt relId="rId8"/>; mit rId6 fiel der Leser stillschweigend auf den Weg ohne Zeichnung zurueck, also ausgerechnet auf den Rueckfall, den die Fixture nicht pruefen soll. Der Waechter ist bewusst bescheiden: jede .pptx-Fixture muss beide OPC-Pflichtteile tragen - die notwendige Bedingung dafuer, dass ein fremdes Office sie ueberhaupt ansieht, und genau sie fehlte; eine Datei mit beiden Teilen und kaputtem Inhalt faengt er nicht, dafuer bleibt die einmalige Messung der Beleg. Sabotage (M131): je Punkt eine Probe (chOff zurueck auf (0,0) -> genau der Test mit verschachtelten Gruppen rot; FindDiagramDrawingTree abgeklemmt -> genau die 2 SmartArt-Tests rot, der Rueckfall-Test gruen; die alte Huelle wieder eingesetzt -> genau 1 von 119 rot, und zwar der Waechter, die zwei Orakel-Tests blieben gruen - der Kern des Befunds: er faengt genau das, was kein anderer Test sehen konnte). Ehrlich offen: SmartArt ohne mitgelieferte Zeichnung wird nicht nachgerechnet (dieselbe Grenze wie im Word-Weg - das Layout zu raten waere schlechter als die benannte Luecke); die Diagramm-Teile stammen weiterhin aus Word, nicht aus PowerPoint; der PPTX-Schreiber kennt nur Balken/Linie/Kreis als Diagrammarten; docProps/thumbnail.jpeg zeigt noch die leere Folie; Platzhalter-Bezug, schemeClr an Formen und Musterseiten-Bindung -> KDX20

#	Punkt	Aufw.	Stand
KDX18	Die Dokument-Ebene an den Klartext-Zielen	M	M1460. Gemessen statt der Zeile geglaubt: die Behauptung „.docx und .odt schreiben sie bereits" stimmte nur zur Haelfte - Kommentare ja (beide), Aenderungsverfolgung nur .docx (ODT schreibt die Endfassung), Abschnitte nur die GEOMETRIE des ERSTEN (beide), und das Feld-Verzeichnis schreibt keiner von beiden (die Klassendoku von FlowExportDocument.Fields sagt es selbst); die Noten der Dokument-Ebene (FlowAnnotations.Notes) wertet ebenfalls keiner aus. Neu die geteilte Auswahlregel Layout/FlowDocumentLevel (Block Anker ueber den Rahmennamen - EINE Stelle, nicht fuenf) und die drei Schreiber auf FlowExportDocument statt Items+Metadata umgestellt. HTML bildet ab: Aenderungen als <ins>/<del> mit Autor im title, Kommentare als <aside> am Dokumentende mit <a class="kommentar-marke"> am Block und verschachtelten Antworten (Ringschluss-fest), der erste Abschnitt als echte @page-Regel samt column-count (CSS Paged Media, keine Attrappe). Markdown bildet ab: die Kommentare als angehaengter Abschnitt „Kommentare" (Inhalt bleibt, Stelle geht verloren); die Aenderungsverfolgung bewusst NICHT - im Text steht die Endfassung, und ein geloeschtes Stueck als rohes HTML einzuschieben wuerde Text hinzufuegen, der in der Endfassung gar nicht steht. Klartext haengt bewusst NICHTS an: das wuerde einer .txt, die jemand weiterverarbeitet, Fremdtext in die Daten schreiben - der Text bleibt der Text, der Bericht nennt den Verlust. Neu erreichbar: oeffentliche Einstiege HtmlWriter.Write/WriteText(LayoutDocument), MarkdownWriter.Write/WriteText(LayoutDocument), FlowTextExport.Write(LayoutDocument) - die Weiche verwarf den Bericht bisher (out _), und ein Bericht, den niemand lesen kann, ist keiner. Verifikation: 13 Tests ueber die echte Weiche DocumentExporter.Export, je mit Gegenbeleg. Sabotage (M131, fuenf Proben, je genau isoliert und zurueckgebaut): Markup-Sicht abgeklemmt -> 2 von 44 rot (die docx/odt-Revisionstests gruen, sie nutzen den Helfer nicht); Faden-Anfaenge -> 2 von 44; @page-Regel -> 2 von 29; Klartext-Bericht -> 5 von 24; Markdown-Bericht -> 3 von 27. Ehrlich offen: die Stellen-Genauigkeit (Marke am Blockende, zweiter Abschnitt, Notennummern) - jetzt KDX19

#	Punkt	Aufw.	Stand
KDX11	Originalteile-Speicher + Rundlauf-Waechter je Format	L	M1459. Zwei Teile: (1) der Originalteile-Speicher - LayoutDocument.OriginalParts/OriginalPartsFormat (neu, generisch: Pfad IM Behaelter -> Bytes, plus das Format), im .kdoc als eigene Ressourcendatei unter resources/original/<format>/<pfad> (KdocFormat.OriginalPartResourceName, dieselbe Aufteilung wie bei den mitgefuehrten Schriften ED23), gepruefte Struktur (KdocValidator.ValidateOriginalParts, dieselbe Form wie bei den Schriften). (2) Der Rundlauf-Waechter - DocxOriginalPartsGuardTests, ein voller Lauf .docx -> .kdoc-Teile -> geladen -> .docx, gegen einen ZWEITEN, unabhaengigen ZIP-Leser geprueft. Die zentrale Entscheidung (mit Begruendung im Code, DocxReader.CarriedOriginalPartPaths): mitgefuehrt wird bewusst NUR docProps/custom.xml (benutzerdefinierte Dokument-Eigenschaften, oft Aktenzeichen/Klassifikation aus einem DMS) - der EINZIGE unbekannte Teil, der nicht veralten kann, weil er auf keinen Absatz-/Tabellen-/Bildindex verweist. docProps/core.xml/app.xml sind ausdruecklich NICHT dabei (sie werden aus dem aktuellen Stand neu erzeugt, ein Original waere ein Widerspruch); customXml/, word/embeddings/, word/theme/*, word/settings.xml bleiben aussen vor (koennen an Struktur gebunden sein, die der Leser wegwirft, z. B. w:sdt-Datenbindung) - der Waechter belegt das ausdruecklich als Luecke, statt es zu verschweigen (KDX11_rundlauf_waechter_customXml_wird_nicht_mitgefuehrt), ebenso den Verlust von word/comments.xml (von DocxToKdoc gar nicht erst uebernommen). Beim Export gehoert zum mitgefuehrten Teil auch seine Sichtbarkeit: [Content_Types].xml-Override + _rels/.rels-Beziehung werden mitgeschrieben, sonst laege er als totes ZIP-Gewicht da. Ein Ziel ausser .docx (z. B. .odt) bekommt den Teil NICHT - geprueft. Ohne mitgefuehrte Teile byte-identisch (kein originalPartsFormat im JSON, kein Ressourcenordner). Verifikation: 6 Tests im .kdoc-Kern (KdocOriginalPartsTests, inkl. dem Fehlerfall „Pfad ohne Format") + 6 im Rundlauf-Waechter, je mit Gegenbeleg (kein Original -> nichts erfunden; ein anderer Pfad ergibt einen anderen Ressourcennamen). Sabotage (M131): PackageBuilder.SetOriginalParts abgeklemmt (per echtem Rot-Test vorgefuehrt, dann zurueckgestellt) -> genau die 2 vom Mitfuehren abhaengigen Tests rot, 3 Gegenbeleg-Tests gruen; die zweite Probe (AddResource-Ablage im .kdoc-Kern) -> genau die vom Ressourcen-Schreiben abhaengigen Tests rot, der byte-identisch-Gegenbeleg gruen. Bewusst nur EIN Format vollstaendig (der Auftrag erlaubte das ausdruecklich): .pptx/.xlsx/.vsdx/.idml/.odt/.odg/EPUB tragen denselben generischen Speicher noch nicht angebunden - das Modell (LayoutDocument.OriginalParts) ist dafuer bereits da, es fehlt je Format der Leser-Allowlist-Eintrag und die Schreiber-Verdrahtung
KDX15	Mehrstufige Liste im Fliesstext	M	M1462. Gemessen statt der Zeile geglaubt, und sie stimmte in BEIDE Richtungen nicht. Zu pessimistisch: die Naht gab es intern laengst - LayoutObjectBlock wird von Flex/Grid als gewoehnlicher In-Fluss-Block genutzt (M235/M236); es fehlte allein der oeffentliche Einstieg. Neu FlowDocument.Object(LayoutObject) (allgemein) und FlowDocument.List(ListFrame) (typisiert). Kein zweiter Listen-Mechanismus: die Fassade reicht durch, Nummerierung, Beschriftung und Einzug bleiben im ListFrame. Zu optimistisch: das .kdoc sicherte Ebenen, Ebeneneinstellungen und ausgezeichnete Laeufe gar nicht - eine mehrstufige Liste wurde beim Speichern still flach; neu KdocList.Levels/LevelIndentStep und KdocListItem.Level/Runs (Schema §8.7), ohne Ebenen/Laeufe byte-identisch. Dazu der zweite Uebersetzungsweg beseitigt: FlowDocument.TranslateLayoutObject hatte fuer eine Liste eine eigene, flache Fassung (ohne Ebenen, Laeufe und Verweis) - LayoutContentExport.TranslateList ist jetzt die eine Stelle. Offen: die Zaehlerstaende einer Fortsetzung (ContinueAfter) ueberstehen den Rundlauf nicht

#	Punkt	Aufw.	Stand
KDX16	Die neuen Fliesstext-Felder an den Importern	M	M1454 + M1462. M1454: LayoutObject.CopyBaseTo kopiert KeepTogether/KeepWithNext (vorher verlor jeder Frame-Typ ausser TextFrame sie beim Klonen); Docx liest w:keepNext/w:keepLines, die Zeilenhoehen-Regel, w:lang/w:highlight je Lauf und den Inline-Hyperlink im gemischten Absatz; Rtf \keep/\keepn/\pagebb; Odf Ueberschriften bis Ebene 9 und die fo:keep-/fo:break-; Epub/Html break-inside; PdfImport /H7-/H9. M1462 - die Sprache je Lauf beim PDF-Import (/Span <</Lang>> BDC ... EMC): die Zeile behauptete, der Content-Stream-Weg kenne keine BDC/EMC-Verfolgung - den Stapel fuehrt der Alt-Text-Weg seit M1456, und TextRun.OperationIndex bindet bereits an dieselbe Zerlegung; es fehlte nur die zweite Auswertung im vorhandenen Durchlauf. Begruendet NICHT gebaut: die Umbruchsteuerung (ein importiertes Dokument hat nachweislich null Fluss-Container - ein Feld, das kein Renderer ausliest) und die Hervorhebung (im Inhaltsstrom ein gewoehnliches gefuelltes Rechteck, von einem Zellhintergrund nicht zu unterscheiden - sie zu erkennen hiesse raten). Grenze: Text aus einem Form-XObject traegt die Nummer des Do-Aufrufs, die Klammer darin ist nicht aufloesbar (wie beim Alt-Text)
KDX17	Fundgrube fuer unvollstaendige Rundlaeufe (Nutzerwunsch)	M	M1461 + M1462. M1461: der Kern - Originaldatei und Befund-Textdatei unter demselben GUID, entdoppelt ueber das Paar aus Inhalts-Pruefsumme und Lueckenart, gedeckelt nach Groesse (2 GB) und Alter (90 Tage), aufgeraeumt beim Ablegen, ab Werk an und ueber KUONEN_PDFTOOLS_FUNDGRUBE=off abschaltbar, nach aussen unsichtbar, in .gitignore und .dockerignore mit Waechter; Andockpunkt Import fuer .docx und Visio, nur Warning. M1462 - der Export: .docx/.odt/.rtf/.fb2 legen bei einem gemeldeten Verlust den Arbeitsstand als gepacktes .kdoc ab, nicht die Zieldatei - die ist daraus herstellbar und enthaelt gerade nicht, worum es geht; deterministisch gepackt, weil die Kennung die Pruefsumme des Inhalts ist (mit der Uhr im ZIP liefe die Entdoppelung leer). Dazu die Hygiene (Nutzerwunsch): scripts/fundgrube.cs listet auf und raeumt, ueber FundgrubeAblage.Auflisten/Leeren; der Ort kommt nur aus FundgrubeEinstellungen, geloescht werden nur Dateien mit Kennungs-Namen, der Ordner nur wenn leer. Ehrlich benannte Grenzen: .txt/.html/.md docken nicht an (ihre Berichte mischen Verlust und blossen Hinweis in deutschen Saetzen - ein Merkmal daraus haenge die Entdoppelung an jede Umformulierung); die uebrigen Zielformate melden nichts Auswertbares; der Befund nennt die Blockart, nicht die Stelle

#	Punkt	Aufw.	Stand
KDX6a	Excel direkt in beide Richtungen (Kern)	XL	M1470. Kassensturz zuerst am Code, und die Zeile stimmte diesmal: .xlsx lief beim Import ueber XlsxToPdf -> gerendertes PDF -> PdfToLayoutDocument, beim Export ueber PdfToXlsx; in src/.../DocumentFormat/ kam Xlsx in keiner Zeile vor. Der Zuschnitt ist bewusst ein anderer als bei Word/Visio/PowerPoint: dort ist die Quelle schon eine Seite aus Rahmen, ein Arbeitsblatt ist dagegen ein Raster aus Werten - Formel, Zahlenformat, Datenpruefung und Fixierung haben im Seitenmodell gar keine Entsprechung. Deshalb entstehen ZWEI Dinge: der Bestand (LayoutDocument.XlsxWorkbook, im .kdoc als xlsxWorkbook, Schema §8.28) und die Ansicht (je Blatt eine gesetzte Seite ueber genau denselben XlsxToPdf-Weg wie das PDF - kein zweiter Zeichenweg; XlsxToPdf.BuildFlow herausgezogen). Zwei Befunde, die nur eine ECHTE Excel-Datei zeigen konnte: (1) Excel legt formula1 einer Datenpruefung in einem mc:AlternateContent ab, die fuer fremde Leser gedachte Fassung steht im mc:Fallback - XlsxReader suchte nur das DIREKTE Kind, jedes von Excel erzeugte Dropdown kam ohne eine einzige Option an; (2) ein deutsch lokalisiertes Excel maskiert US-Formatcodes zeichenweise als Literale (yyyy-mm-dd -> \y\y\y\y\-mm\-\d\d) und zeigt die Zelle dann selbst als yyyy-03-dd - unser Leser lag RICHTIG, meine erste Fixture war falsch (in der Fixture-Doku festgehalten). Neu gebaut: XlsxCell.NumberFormat/XlsxWriteCell.NumberFormat samt der builtin-Tabelle nach ISO/IEC 29500 §18.8.30 (Excel schreibt fuer builtin-Kennungen KEINEN numFmt-Eintrag - ohne die Tabelle waere jedes Standardformat still verloren) und der Stiltabelle CellStyleTable im Schreiber (je Kombination aus Fett-Schnitt und Format EIN cellXfs-Eintrag statt der vier fest verdrahteten; die ersten vier Indizes bleiben belegt, ein Test haelt die bisherige Stilvorlage woertlich fest). XlsxToKdoc/KdocToXlsx/XlsxKdocReport, Layout/XlsxWorkbookInfo, KdocPartJson.Xlsx, beide Weichen (DocumentConverter.ToKdocDocument, DocumentExporter.Export) samt Fundgrube-Andockung. Orakel, beide Richtungen und beide Werkzeuge: die Fixture stammt aus Excel 16 (COM); die von uns GESCHRIEBENE Mappe und der volle Rundlauf .xlsx -> .kdoc -> .xlsx wurden in Excel nachgemessen (Blaetter, Fett, Format 0.00 -> 1.50, Formel SUM(C2:C3) -> 3.75, Datum JJJJ-MM-TT -> 2024-03-15, Verbund $A$7:$B$7, FreezePanes/SplitRow=1, Autofilter, Dropdown Ja;Nein, bedingte Formatierung, benannter Bereich) und von LibreOffice konvertiert (Rueckgabewert 0). Dazu XlsxFixtureHuelleGuardTests gegen eine Fixture ohne die OPC-Pflichtteile. Sabotage (M131, fuenf Proben, je genau isoliert und zurueckgebaut): Nachfahren-Suche der Datenpruefung -> genau 1 von 152 rot; Zahlenformat im Schreiber -> genau 3 von 158 (beide Gegenbelege gruen); builtin-Tabelle -> genau 1; Import-Weiche -> genau 3 von 165; Export-Weiche -> genau 1; JSON-Schreiben des Datenteils -> genau 1. Offen: KDX6b (Pivot, Zellanker, Diagramm-Quellbezug, Zellformat beim Schreiben, Originalteile)

v4-AP · Acrobat Pro

Gruppe	Worum es geht	Aufw.	Stand
AP1 · Barrierefreiheit REPARIEREN	Auto-Tagging eines fremden, ungetaggten PDF (Acrobat „Autotag Document“) + Reading-Order-Werkzeug.	XL	Accessibility/AutoTagger repariert fremde PDFs auf Byte-Ebene, MCP auto_tag_pdf. Das Reading-Order-Werkzeug ist M1373 (ED41): Editing/StructureTreeEditor (Rolle/Alternativtext/Kinder-Reihenfolge eines vorhandenen /StructTreeRoot an Ort und Stelle aendern) + Playground/Model/AccessibilityTreeView (Baum->flache Zeilen) + der Reiter „Tag-Baum / Leseordnung" im Editor-Overlay

Gruppe	Worum es geht	Aufw.	Stand
AP2 · Prepare Form (Feld-Autoerkennung)	Formularfelder auf einem flachen PDF/Scan automatisch erkennen + anklickbar anlegen (Text/Checkbox/Radio).	L	Extraction/FormFieldDetector (erkennen) + FormComposition/FormFieldAutoPlacer (anlegen ueber den vorhandenen Formular-Weg), MCP detect_form_fields. Was nicht sicher zuzuordnen ist, wird benannt statt geraten. Grenzen: reiner Bild-Scan braeuchte Bilderkennung; Optionsgruppen und Auswahllisten sind aus der Geometrie allein nicht erschliessbar
AP3 · Bild in-place bearbeiten	Ein Bild in einem fremden PDF anklicken -> verschieben/skalieren/ersetzen, ohne Import-Rundlauf.	L	Editing/PdfImageEditor (auflisten/ersetzen/verschieben/loeschen) + MCP edit_image. Grenzen: nur unmittelbar auf der Seite gezeichnete Bilder - Form-XObject und Inline-Bild werden benannt abgelehnt; MoveImage erkennt die einfache q cm Do Q-Form
AP4 · Messwerkzeuge	M1325 - Annotations/RectilinearMeasurement (rechnen) + /Measure /Subtype /RL an den Formen, MCP add_measurement. Grenze: nur gleichmaessiger Massstab	M	
AP5 · Transparenz-Flattener	M1321 - Editing/PdfTransparencyFlattener, MCP flatten_transparency. Grenze: die Seite wird ganz gerastert (keine vektorielle Zerlegung)	L	
AP6 · Rich Media / 3D	Video/Audio als abspielbare RichMedia-Annotation; U3D/PRC-3D einbetten + 3D-Ansichten.	XL	PdfRichMediaAnnotator + Pdf3DAnnotator + AnnotationPosterAppearance, MCP add_rich_media/add_3d_annotation. Ehrlich: die Bibliothek spielt nichts ab und rendert kein 3D - sie bettet ein; /RichMedia ist eine Adobe-Erweiterung ausserhalb von ISO 32000, in PDF/A verboten
AP7 · Editor: fremdes PDF IN-PLACE editieren	Im visuellen Editor ins gerenderte Fremd-PDF klicken und Text/Bild direkt aendern (byte-schonend), statt Import->Re-Render (ED14).	XL	M1373 (ED42): Playground/Components/Editor/ForeignPdfEditorPanel - dasselbe InPlaceEditor (kein zweiter In-Place-Weg) jetzt auch als Overlay der grossen Editor-Seite (Menue Datei -> „Fremdes PDF an Ort und Stelle bearbeiten..."), mit dem Reiter „Tag-Baum/Leseordnung" (ED41) daneben. Grenzen: kein Reflow, keine Form-XObjects (unveraendert, aus InPlaceEditor)

v4-LID · InDesign

#	InDesign-Funktion	Stand bei dir	Aufw.
LID1	Objekt-Effekte ausser Schlagschatten: innerer Schatten, Schein (aussen/innen), abgeflachte Kante/Relief, Satin, weiche Kante (Feather)	Layout/ObjectEffects (die weiche Kante ueber eine echte PDF-Weichmaske; die uebrigen vier sind eine rechteckige Naeherung, nicht silhouettentreu - in der Klassendoku benannt; noch nicht im .kdoc gesichert)	L
LID2	Pathfinder + zusammengesetzte Pfade: Vereinen/Subtrahieren/Schnittmenge/Ausschluss; Compound Paths	(Layout/Pathfinder, ShapeKind.CompoundPath)	M

#	InDesign-Funktion	Stand bei dir	Aufw.
LID3	Text-Variablen <var:name> (Dateiname/Version/Kapitel/benutzerdefiniert, TextVariables.Build) M1284; laufender Kolumnentitel (FlowDocument.BuildWithRunningHeader) M1345 - letzte Seitenzahl war ueber FlowDocument.PageNumbers ({total}) bereits erledigt	M1284/M1345	M
LID4	Grundlinienraster + optischer Randausgleich (haengende Interpunktion)		M
LID5	Editor: Ausrichten & Verteilen + Duplizieren-und-versetzen (Step and Repeat)	EditorAlignDistribute/EditorStepAndRepeat + Oberflaeche. Grenze: gerechnet wird auf dem ungedrehten Rahmen, eine Gruppe zaehlt als ein Rechteck	M
LID6	Liquid Layout / alternative Layouts (adaptive Regeln fuer mehrere Seitengroessen)	M1347 (Layout/LiquidLayout: Skalieren, Neu zentrieren, Objektbasiert) + M1430 die hilfslinienbasierte Regel. Dafuer zuerst die Bibliotheksfaehigkeit: LayoutGuide + LayoutPage.Guides (Ausrichtung, Lage, Name, Kennzeichen „fluessig"), im .kdoc nur geschrieben, wenn vorhanden, nie gezeichnet. Die Regel folgt einem Nahtmodell (nicht Adobes Algorithmus, das steht so in der Klassendoku): jede fluessige Linie nimmt ihren gleichen Anteil der Groessenaenderung auf, ein durchschnittenes Objekt waechst, ohne fluessige Linie auf einer Achse gilt Neu zentrieren; eine Linie genau auf einer Objektkante schneidet nicht. LiquidLayout.ApplyToDocument erhaelt Verkettungen; MCP liquid_layout; im Editor sind die Lineal-Hilfslinien jetzt je Seite im Modell (vorher global und nach dem Neuladen weg), als fluessig markierbar und das ganze Dokument per Liquid Layout umrechenbar. Gemeldet statt still: Doppelseiten, Anmerkungen und Musterseiten werden nicht umgerechnet; gleiche Anteile je Naht (keine Gewichtung), Schrumpfen kappt bei 0, Objektdrehung fliesst nicht ein	L

v4-LWD · Word (.docx)

#	Word-Funktion	Stand bei dir	Aufw.
LWD1	Aenderungsverfolgung LESEN: Accept-Changes-Endfassung (w:ins/w:del) M1289 + Markup-Ansicht (Einfuegung gruen, Loeschung rot durchgestrichen) + Kommentare (word/comments.xml in den Lesebericht) - ehrlich benannte Grenze: Aenderungsverfolgung und Kommentare schreiben nicht umgesetzt	M1289+	L
LWD2	Fussnoten in .docx LESEN (w:footnoteReference) M1286 + SCHREIBEN (DocxWriter legt word/footnotes.xml an) + Endnoten lesen (ans Dokumentende, eigener Zaehler)	M1286+	M
LWD3	Felder: w:fldSimple M1288 + komplexe Felder (w:fldChar Begin/Separate/End) - je der zwischengespeicherte Wert, nicht der Feldcode; die Felder werden nicht neu berechnet (ehrlich benannte Grenze)	M1288+	M
LWD4	Kopf-/Fusszeilen + Abschnitte beim LESEN (verschiedene Seitengroessen/Spalten je Abschnitt)	w:sectPr mit headerReference/footerReference, w:pgSz und w:cols; gestempelt ueber den vorhandenen HeaderFooterStamper	M

#	Word-Funktion	Stand bei dir	Aufw.
LWD5	Formeln (OMML/Office-Math) lesen/schreiben	Docx/OmmlToLatex (lesen, ueber den vorhandenen Formelsatz gesetzt) + Docx/LatexToOmml (schreiben, M1356-Rest): Brueche, Wurzeln (auch mit Grad), Hoch-/Tiefstellung, mitwachsende Klammern, grosse Operatoren mit Grenzen (m:nary, Summe/Produkt/Integral - undOvr bzw. subSup je nach \int gegen \sum) und griechische Buchstaben/Symbole als Roh-Unicode. Ohne Mathe-Schrift bzw. bei Akzenten/Ueber-Unterstrichen/Matrizen wird die Formel gemeldet statt geraten (kein Halbergebnis)	L
LWD6	Textrahmen/Formen/SmartArt/Diagramme	M1356 (Textrahmen als Absatz) + M1431: neues gemeinsames Modul DrawingMl (Vorlage- und freie Geometrie, Fuellung, Kontur, Themenfarben samt lumMod/lumOff/tint/shade im linearen Raum, Transformation, Textkoerper, Diagrammleser) - PowerPoint und Excel lesen darueber, kein dritter Parser. Word-Formen, Textrahmen (Rahmen + formatierter Text, Innenabstand, senkrechte Ausrichtung) und Gruppen werden gezeichnet; wp:inline fliesst im Text, wp:anchor sitzt absolut nach positionH/positionV (page/margin/column/paragraph, posOffset/align/pctPos), Umfluss ueber FlowDocument.FloatObject. SmartArt ueber die von Word abgelegte vorgerenderte Zeichnung (word/diagrams/drawingN.xml) - kein nachgebauter Layout-Algorithmus; ohne Zeichnung die Knotentexte als Liste nach der Hierarchie (DOCX_DIAGRAM_TEXT_ONLY). Diagramme als natives ChartFrame. Sollwerte aus Microsoft Word 16 (Fixtures per COM erzeugt, PDF-Export vermessen) und LibreOffice. Satzspiegel aus w:pgMar statt fest. M1432 (die M1431-Grenzen geschlossen): alle 187 Vorlageformen ueber einen Formelrechner nach ECMA-376 §20.1.9 samt Textrechteck, freie Konturen mit mehreren Pfaden; Text im Textrechteck mit Absatzformat, Aufzaehlung, normAutofit/spAutoFit, senkrechter Schreibrichtung, Tabellen und Bildern; Konturen mit Deckkraft, allen Strichmustern, Pfeilspitzen, Mehrfachlinie; Fuellungen mit 54 Mustern, Bild, Pfad- und mehrstufigem Verlauf; Schatten/Schein/weiche Kante/Spiegelung/Relief silhouettentreu; verankerte Formen hinter dem Text, Umfluss eng und mehrspaltig; alle Diagrammarten ausser Oberflaechen samt Achsentitel, Datenbeschriftungen, Gitter, Achsengrenzen, Log-Achse, Zahlenformaten; .docx direkt nach .kdoc (DocxToKdoc, Schema §8.21/§8.22); fette Standardschrift mit richtigem Vorschub im Rasterizer. Orakel Word 16 und PowerPoint 16 (COM), PDFium. Bleibt genaehert (gemeldet): Kurven als Strecken (0,05 pt), Anker am Zeichen, wrapThrough wie wrapTight, hinter dem Text nur ohne Umfluss, Anfasser/extrusionOk nicht ausgewertet, perspektivische Kamera/Material/Reliefprofile; Excel mit Office-Vorgabethema, PowerPoint-Writer nur Balken/Linie/Torte und Vorlageformen teils als rect, Oberflaechendiagramme, ofPie/3D genaehert, Beschriftungen in Helvetica ohne manualLayout; im .kdoc fehlen Kopf-/Fusszeilen- und Fussnotentext; fette Schnitte mit duennem Strich, Times ohne Serifen im Rasterizer	

v4-LXL · Excel (.xlsx)

#	Excel-Funktion	Stand bei dir	Aufw.
LXL1	Formel-Auswertung (numerischer Rechenkern XlsxFormula: SUM/AVERAGE/MIN/MAX/COUNT/IF/ROUND/ABS/VLOOKUP + Arithmetik/Vergleiche/Bereiche) + Formeln schreiben (<f>, M1291); Text-Funktionen (LEFT/RIGHT/MID/LEN/CONCATENATE/TRIM/UPPER/LOWER/TEXT), Datums-Arithmetik (DATE/YEAR/MONTH/DAY/EDATE/DATEDIF, Excel-1900-System inkl. Schaltjahr-1900-Bug), benannte Bereiche in Formeln (XlsxDefinedName aufloesbar) und die Neuberechnungs-Kette (XlsxFormulaEngine.RecalculateAll — Abhaengigkeitsreihenfolge + Ringbezug-Erkennung) — ehrlich benannte Grenzen: kein TODAY()/NOW() (nicht deterministisch), TEXT nur einfache Nachkommastellen-Muster (keine Datumsformate)	M1291+	XL
LXL2	Zellformatierung lesen + rendern: Fuellfarbe/Fett/Schriftfarbe (M1291), Zellrahmen (duenn/mittel/dick) + Ausrichtung (waagerecht/senkrecht) nach Zellstil (M1291-Rest)		L
LXL3	Bedingte Formatierung (cellIs/colorScale/dataBar/iconSet read/write + im xlsx->pdf angewandt) + Datenpruefung (list-Dropdowns + Zahl-/Datums-/Textlaengen-Bedingungen, read/write) - komplett (M1293, M1358)	M1293/M1358	L
LXL4	Diagramme LESEN (Balken/Linie/Torte/Flaeche/Streu/Ring - Zeichnungs-Kette -> chart-Teil) + SCHREIBEN (XlsxWriteSheet.AddChart, eine Reihe je Diagramm) + als natives ChartFrame in xlsx->pdf gerendert (Flaeche/Ring als Naeherung ueber Linie/Torte, ehrlich benannt)		L
LXL5	Benannte Bereiche (definedName) + Fensterfixierung (pane) + Autofilter (autoFilter) read/write (M1294) + Pivot-Tabellen lesen (Quellbereich, Zeilen-/Spalten-/Wertfelder) und im PDF als zur Laufzeit aus den Rohdaten gerechnete Kreuztabelle darstellen — nur sum/count/average als echte Aggregation, keine berechneten Felder, keine verschachtelten Mehrfach-Zeilenfelder (M1360)	M1294/M1360	M (Nische)
LXL6	Bilder im Blatt LESEN (Zeichnungs-Kette -> xl/media) + in xlsx->pdf gerendert (M1295) + SCHREIBEN (XlsxWriteSheet.AddImage - xl/drawings + xl/media + rels; Diagramme und Bilder teilen sich EINE drawingN.xml je Blatt, ihre Beziehungen einen rId-Raum) - ehrlich benannte v1-Grenzen: nur PNG (die Bytes werden durchgereicht, nicht geprueft oder umgewandelt), nur der Anker-Typ oneCellAnchor (keine Groessenanpassung an den Zellinhalt), keine Formen im Zeichnungsteil		M (Nische)

v4-LFO · Apache FOP (XSL-FO)

#	FOP-Funktion	Stand bei dir	Aufw.
LFO1	fo:marker / retrieve-marker (laufende Kolumnentitel aus dem Inhalt)	M1312 (Fo/FoContext + HeaderFooterOptions.ResolvePerPage): der Kolumnentitel wird je Seite aus dem Inhalt aufgeloest, ueber den vorhandenen HeaderFooterStamper. Grenzen: retrieve-position als first/last auf der Seite, keine retrieve-boundary-Feinheiten, Markerinhalt als Klartext	M
LFO2	Zell-Spans (number-columns-spanned / number-rows-spanned) + background-color je Zelle	M1281	M
LFO3	fo:leader M1285; fo:float + echte fo:footnote M1312 (ueber die vorhandene Umflussmechanik FloatToTop/FloatToSide M234/M1022 und ParagraphWithFootnotes M991 - nichts neu gebaut). Grenzen: Float-Hoehe per Messlauf, before-Float schiebt nur Folgeinhalt, Fussnote in einer Change Bar wird gemeldet	M1312	M

#	FOP-Funktion	Stand bei dir	Aufw.
LFO4	fo:bookmark-tree (explizite PDF-Lesezeichen) + Change Bars + fo:index	M1312: Lesezeichenbaum ueber FlowDocument.AddOutline/AddChild, Change Bars ueber BoxFrame.BorderLeft (kein neuer Zeichenweg), Register ueber IndexMarker/BuildWithIndexes. Grenzen: unaufloesbares Ziel wird gemeldet, Change Bar am Block statt je Zeile, Index ODER Fussnoten	M
LFO5	Weitere FOP-Ausgaben (PostScript/PCL/AFP/PNG/TIFF aus demselben FO)	M1312/M1426 (Fo/FoOutput + MCP fo_to_png/fo_to_tiff/fo_to_ps/fo_to_pcl): PNG je Seite, mehrseitiges TIFF, PostScript Level 2, HP PCL5 monochrome Rastergrafik (Print-Stream fuer Direktdruck, rundlauffaehig ueber PclToPdf). Multi-page-sequence-Zusammenfuehrung, fo:declarations-Metadaten und PDF/UA-Tagging nach goldenem Standard. AFP als IBM-Grossrechner-Format bewusst nicht im Kern	L

v4-LTX · LaTeX (pdflatex/xelatex)

#	Punkt	Stand	Aufw.
LTX3	Theorem-Umgebungen (\newtheorem) M1283; \substack und kommutative Diagramme (amscd \begin{CD} mit @>>>/@<<</@VVV/@AAA/@=/@|/@. samt Pfeilbeschriftung) M1353 - beide ueber den vorhandenen Matrix-Baustein, kein zweiter Satzweg. Grenze: die Beschriftung steht als Hoch-/Tiefstellung am Pfeil, nicht exakt darueber zentriert	M1283/M1353	M
LTX4	book/report-Klasse: \part/\chapter + Nummerierung (Kapitel.Abschnitt)	M1282	M

#	Vorbild	Punkt	Stand	Aufw.
LTX5	Beamer (Praesentationsklasse)	M1355/M1359/M1420: \documentclass{beamer} setzt Folien - eine Seite je frame, Folienformat 128×96 mm bzw. 160×90 mm bei aspectratio=169, \frametitle und der Titel hinter \begin{frame}{...}. Overlays wirken wirklich: \pause rollt die Folie auf der Token-Ebene zu Zwischenfolien aus (M1359), dazu die Bereiche \item<2->, \only, \uncover, \visible, \onslide und \alert (M1420) - \alert blendet dabei nicht aus, es hebt hervor und steht auf jeder Zwischenfolie. \usetheme setzt eine Fussleiste, \setbeamertemplate bildet footline/headline/frametitle/navigation symbols ab. Gemeldet statt still: die Plus-Formen <+->/<.->, die Aktionsangabe <2| alert@2>, \setbeamercolor/-font, und \uncover blendet aus statt Platz frei zu lassen	M1355	L
LTX6	fontspec/XeLaTeX + microtype	M1352: \setmainfont/\setsansfont/\setmonofont (auch BoldFont=/ItalicFont=) binden eine Systemschrift ein - opt-in ueber LatexConversionOptions.FontDirectories, sonst wird das Dateisystem nicht angefasst und der Bericht sagt es; eine nicht gefundene Familie wird benannt statt still ersetzt. microtype: der Randausgleich (Protrusion) wirkt, die Font-Expansion ist nicht umgesetzt (gemeldet)	M1352	M

#	Vorbild	Punkt	Stand	Aufw.
LTX1	TikZ/PGF	M1354/M1357/M1358/M1420/M1428: Koordinaten, Pfade, Knoten, scope/\foreach, positioning, Baeume, Matrizen, pgfplots, Funktionsplots und logarithmische Achsen - M1428 dazu: eigene Achsengrenzen xmin/xmax und Teilstrichlisten xtick (auch auf der log-Achse, durch denselben log10-Weg), parametrische Plots ({cos(t)},{sin(t)}) ueber denselben Ausdrucksrechner und damit dieselbe Winkeleinheit (Grad), Balken ybar/xbar, Fehlerbalken und Marker, bei den Baeumen grow= in Grad, level distance=/sibling distance= und der rechtwinklige edge from parent fork down|up|left|right, und schliesslich ein eigener edge from parent path={...}: die Platzhalter \tikzparentnode/\tikzchildnode werden als benannte Knoten gebunden und der Ausdruck laeuft durch den vorhandenen Pfadleser - kein zweiter Auswerter; dabei fielen |-/-| und relative Bezier-Kontrollpunkte +(dx,dy) fuer jeden Pfad ab. Gemeldet statt geraten: eine Ankerangabe (A.south) trifft die Knotenmitte; ein Punkt ausserhalb der Achsengrenzen faellt ganz weg (die Linie bekommt eine Luecke, statt bis zum Rand gezogen zu werden); Flaechendiagramme (fill between, stack plots), gruppierte Balken, externe Datendateien, grow' und Abstaende je Ebene	M1428	XL
LTX2	BibTeX/biblatex (.bib lesen + Zitierstile)	M1428: Latex/BibTexParser (alle vier Namensformen, @string mit #-Verkettung, verschachtelte Klammern) + BibliographyFormatter, Sortierung nach Autor/Jahr (nyt/nty/none) und die Suffixe 2020a/2020b (M1420) - M1428 dazu: ein echter .bst-Interpreter (Stapelmaschine mit ENTRY/FUNCTION/READ/ITERATE/SORT, den $-Operationen und der Formatsprache von format.name$). Angebunden wie \includegraphics: die .bst wird nur gelesen, wenn ein Basisverzeichnis gesetzt ist und die Datei danebenliegt; ein abbrechender Stil faellt gemeldet auf die zwei eingebauten zurueck, ein \bibliographystyle{plain} ohne beiliegende plain.bst ist dagegen der Normalfall und wird bewusst nicht gemeldet. Fremdverifiziert: die vier Originalstile plain/alpha/abbrv/unsrt laufen unveraendert aus TeX Live durch und treffen die Ausgabe eines echten BibTeX 0.99d auf dieselbe .bib - im Testlauf laeuft weder TeX noch Docker, beides liegt als Fixture bei. Grenze: die Kollation ist genaehert statt sprachspezifisch nach ISO 14651	M1428	L

v4-LHT · HTML/CSS (Browser-Rendering)

#	Browser-Funktion	Stand bei dir	Aufw.
LHT1	CSS Flexbox (display:flex, flex-grow/wrap/order/align-self)	M237/M643 (war faelschlich als Luecke gefuehrt)	L
LHT2	CSS Grid (display:grid, auto-fill)	M-Grid (war faelschlich als Luecke gefuehrt)	L
LHT5	@media (Typ + Breiten-Features min/max-width) M651; position:sticky als In-Fluss gerendert (Scroll ist kein PDF-Konzept)	M651 (war faelschlich als Luecke gefuehrt)	M
LHT6	CSS Custom Properties (--x/var()) + calc()/min/max/clamp	M887/M888 (war faelschlich als Luecke gefuehrt)	S
LHT4	JavaScript — deterministischer Browser-Ladelauf (M1465), Modul-Randfaelle, lebende Import-Bindungen, Klassenfelder, Anweisungen (M1466), Pfeilfunktionen, Template-Literale, Destrukturierung, Spread, Generatoren samt Iterator-Protokoll, async function*/for await, echtes Symbol, String.prototype vollstaendig, regulaere Ausdruecke nach ECMAScript mit u/v-Flag und allen Unicode-Eigenschaften, normalize ueber eigene UCD-Daten, eval (M1467)	M1467	M

#	Browser-Funktion	Stand bei dir	Aufw.
LHT3	CSS Transforms und Animationen — 2D/3D samt transform-origin, Animationen als Standbild nach CSS-Zeitsemantik (M1466); echte Perspektive (M1467/M1468): Hintergrund, Rahmen, Rundung, Schatten, Umriss, clip-path, Text als projizierter Glyphenumriss, Hintergrundbilder adaptiv gekachelt (<= 0,3 pt), Nahebenen-Beschnitt, perspective-origin, Inline-Kaesten, preserve-3d als gemeinsame Szene mit BSP-Tiefenordnung und Aufteilung sich schneidender Ebenen, backface-visibility in Szenen-Koordinaten; gegen Edge quantitativ geprueft. Grenzen: Inline-Kaesten innen tangential, der Edge-Pixelvergleich trifft die Farbe je Quadrant, nicht die Grenzlinie pixelgenau	M1468	M

v4-LPP · PowerPoint (.pptx)

#	PowerPoint-Funktion	Stand bei dir	Aufw.
LPP2	Tabellen im Foliensatz (graphicFrame/tbl) lesen + schreiben		M
LPP5	Notizen (notesSlide) + Folien-Kommentare lesen/schreiben	(Notizen ohne Folienbild-Platzhalter, klassisches p:cm-Kommentarmodell, beides im pptx->pptx-Rundlauf erhalten)	S
LPP6	Themen (theme1.xml: Farb- und Schriftschema) statt Standard-14-Abbildung	(Farbschema aufgeloest - schemeClr in Laeufen, explizite srgbClr hat weiter Vorrang; Schriftschema gelesen, noch nicht auf den Textsatz angewandt)	M
LPP7	Bild-Zuschnitt (blipFill/srcRect), Gruppen (grpSp), Drehung/Spiegelung	(Zuschnitt/Drehung/Spiegelung gelesen + im PDF gezeichnet, Bildpunkt-verifiziert; Gruppen beim Lesen auf absolute Folienkoordinaten aufgeloest - der Rundlauf schreibt sie flach zurueck)	M
LPP8	Diagramme + SmartArt (c:chart, DrawingML-Diagramme) — die Art ueberlebt den Rundlauf	M1474: PptxChartKind (drei Werte) ist ersatzlos entfallen, PptxChartData.Kind IST jetzt Layout.ChartKind - dieselbe Aufzaehlung wie ChartFrame und der .kdoc-Weg; zwei Enums fuer dieselbe Sache waren zwei Quellen einer Wahrheit und liefen auseinander: ein Ringdiagramm kam als Torte zurueck, ein Flaechen-, Punkt-, Blasen-, Netz- oder Kursdiagramm als Linie. Alle neun Arten werden jetzt als ihr eigenes <c:xxxChart> geschrieben; Punkt und Blase mit <c:xVal>/<c:yVal> statt <c:cat>/<c:val>. Der Fund, den erst das Orakel zeigte: PowerPoint 16 lehnte 7 von 9 geschriebenen Dateien als „beschaedigt" (HRESULT 0x80070570) ab - jede Art ausser Torte und Ring verlangt zwei <c:axId>-Verweise UND die Achsen in der plotArea; betroffen war auch das seit M1420 geschriebene Saeulen- und Liniendiagramm, der eigene Rundlauf ging darueber hinweg. Belegt mit beiden Orakeln (PowerPoint 16 liest alle neun, LibreOffice konvertiert alle neun mit Rueckgabewert 0). SmartArt wird seit KDX5 gezeichnet (die drei Texte, die noch das Gegenteil behaupteten, sind berichtigt). Dabei aufgefallen: Tabellen fielen frueher STILL aus dem Rundlauf, sie werden seit M1420 gemeldet. Der Rest steht als LPP10	L (Nische)
LPP9	Uebergaenge/Animationen - nur fuer den pptx->pptx-Rundlauf sinnvoll (im PDF nicht darstellbar)	(PptxPresentationWriter: p:transition/p:timing roh erhalten; PptxToPdf meldet sie ehrlich statt sie stillschweigend zu uebergehen)	M (Nische)

v4-LVS · Visio — in M1437 geschlossen (die Gruppe ist fertig)

#	Punkt	Stand	Aufw.
LVS18	Feinheiten: Bilder in binaeren .vsd (Fassung 11, auch die Stromform in Mastern und PNG aus OLE-Objekten); Verlaeufe nach Visio 2013 in allen Richtungen und Arten 1..13, Deckkraft je Stopp, Mitdrehen mit der Form, weicher silhouettentreuer Schatten — in PDF, .kdoc und Rueckexport; Themen-Effekte (Schein, weiche Kante, abgeflachte Kante, Spiegelung, Themen-Schatten) ueber den Effekt-Zeichner aus M1432, im .kdoc gesichert und als Effektzellen zurueckgeschrieben; Stilvorlagen im Zeichnungsmassstab der nutzenden Seite. Benannte Grenzen: Bilder in .vsd 5/6 ungeprueft (keine freie Pruefdatei), Bildversatz ungleich 0 ungetestet, OLE ohne Vorschau gemeldet; Deckkraft-Rampe linear (PowerPoint zeichnet sie nicht linear, fuer Visio nicht belegt); BevelBottom/Tiefe/Kontur/Material/Licht ausdruecklicher Zellen nicht ausgewertet, Licht fest von links oben; Spiegelung ohne Text, Bild-Shapes ohne Effekte; Spiegelung und 3D-Tiefe im .kdoc gemeldet statt gesichert	M1437, gegen LibreOffice (Lage, Bilder) und PowerPoint 16 (Effekte)	L

v4-LVS · Visio — in M1436 geschlossene Scheiben

#	Punkt	Stand	Aufw.
LVS8b	Darstellung, Rest: eingebettete Bilder (Raster ueber RasterImage.Parse, EMF/WMF als Vektoren ueber Metafile) auf das Shape beschnitten, im .kdoc als ImageFrame bzw. PdfPageFrame, beim Export als Fremdobjekt unter visio/media, auch aus .vdx; Fuellmuster 2..40 (Schraffuren und alte Verlaeufe), Verlaeufe nach Visio 2013, Transparenz von Flaeche und Linie, Schatten — in PDF, .kdoc (fillHatch, Verlauf, Deckkraft, Schlagschatten) und zurueck	M1436, gegen LibreOffice (wo es sie auswertet; sonst Office-Konvention, benannt)	L
LVS10c	Themen und Ebenen: mehrere Themen-Teile, Farbschemata und Varianten je Seite, Farbabwandlungen ueber DrawingMlColors, Stil-Matrix samt Themen-Strichstaerke, Linienfarbe ueber die ganze Stilkette, Verbinder-Format des Themas; Ebenen im Objektmodell (LayoutObject.LayerNames, PdfLayer mit „Gesperrt"), im .kdoc (layers), im Visio-Rundlauf und als Editor-Palette (sichtbar/druckbar/gesperrt)	M1436	M
LVS14b	.vsd der Fassungen 5 und 6 (Visio 5 bis 2002) — dazu zwei Fehler im Fassung-11-Weg behoben: die Linienzeile wurde ein Byte versetzt gelesen (jede Linie unsichtbar), und der Zeichnungsmassstab fehlte	M1436, Pruefdateien aus dem libvisio-Bestand (MPL-2.0)	M
LVS19	Zeichnungsmassstab im .vsdx: PageScale/DrawingScale rechnen Lage, Groesse, Geometrie (samt absoluten NURBS/POLYLINE-Punkten), Textblock, Bildlage und Seitengroesse in Papiereinheiten um — eine Regel VisioMassstab fuer .vsd und .vsdx; dabei behoben, dass eine Seite ueber 500 Zeichnungszoll still auf 8,5 x 11 zurueckfiel	M1436, gegen LibreOffice	S

v4-LVS · Visio — in M1435 geschlossene Scheiben

#	Punkt	Stand	Aufw.
LVS7	Freie Kurven exakt: NURBSTo (polynomial exakt ueber die Hermite-Form je Knotenspanne; rational oder hoehergradig adaptiv mit gemessener Abweichung und VSDX_GEOMETRIE_GENAEHERT), PolylineTo, SplineStart/SplineKnot, InfiniteLine an der Seite beschnitten	M1435, gegen LibreOffice	M
LVS9b	Text, Rest: fette Standard-14-Schnitte im eigenen Rasterizer wirklich fett (der Leser war richtig, mager war das Bild); Absatzformat je Absatz, leere Absaetze und Visio-Grundlinien im .kdoc; die genannte Schrift opt-in eingebettet ueber VsdxConversionOptions.FontDirectories (wie fontspec im LaTeX-Weg), ohne Ordner Standard-14 mit VSDX_SCHRIFT_ERSETZT	M1435	M
LVS10b	Ebenen: Layer (Name/Visible/Print) und LayerMember; ein Shape auf nur unsichtbaren Ebenen wird nicht gezeichnet, auf nur nicht druckbaren nicht ins PDF (die Druckausgabe); gemeldet	M1435 (LibreOffice wertet das Druck-Kennzeichen nicht aus, es gilt Visios Zellbeschreibung)	M
LVS11	Visio-Gruppen erhalten: im Lesemodell VsdxPage.Groups, im .kdoc verschachtelte ObjectGroups mit dem aus den Kind-Formeln gelesenen Groessenverhalten, beim Export Type='Group' mit den Formeln, die Visio selbst schreibt; nicht eindeutige Formeln bleiben proportional und werden gemeldet (VSDX_GRUPPENVERHALTEN_UNKLAR)	M1435, LibreOffice zeichnet den Rueckweg an denselben Stellen	M
LVS12	Export vollstaendig: ein Verbinder geht als 1-D-Verbinder mit _WALKGLUE/<Connect> an die benannten Formen; ein zusammengesetzter Pfad mehrere Geometry-Abschnitte, gerade-ungerade gefuellt (eine Regel VsdxGeometrien.Verbund fuer PDF und .kdoc)	M1435	M
LVS13	Kurven im .kdoc: neu ShapeKind.Path mit ShapePathSegment (M/L/C/Z) und Even-Odd, im .kdoc als path/evenOdd, gezeichnet mit echten Kurven, Export als CubBezTo	M1435	M
LVS14	Alte Formate: .vdx (XML, in die .vsdx-Zellform uebersetzt) und .vsd Fassung 11 (OLE-Verbunddokument, LZ77, Chunks) — beide laufen durch den vorhandenen VsdxReader; in DocumentConverter und den MCP-Werkzeugen erreichbar	M1435, Pruefdateien aus dem libvisio-Bestand (MPL-2.0), Sollwerte aus LibreOffice	L
LVS15	Feinabgleich: graue Titelbalken sind richtig (Visios eigenes Vorschaubild zeigt Grau, LibreOffice Weiss — es gilt Visio); Themen-Linienfarbe ueber QuickStyleLineColor; Linienenden wachsen linear mit der Strichstaerke (an LibreOffice gemessen); Themed-Zeichenfarbe wird dk1	M1435	S

v4-LVS · Visio — in M1434 geschlossene Scheiben

#	Punkt	Stand	Aufw.
LVS8a	Strichmuster und Linienenden: LinePattern 2..23 in Vielfachen der Strichstaerke, die 45 Visio-Linienenden (Pfeile, Kraehenfuesse) in beide Richtungen — Modell, PDF, .kdoc (startLineEnd/endLineEnd), Export	M1434, Sollwerte aus LibreOffice	M
LVS9a	Text originalgetreu: cp/pp/tp/fld, Zeichen- und Absatzformate zellweise geerbt, Textblock mit Raendern, senkrechter Ausrichtung, Hintergrund und Drehung, Umbruch ueber TextLayout; im .kdoc ein formatierter Rahmen je Textblock, Rueckweg mit cp/pp	M1434, Grundlinien gegen LibreOffice im Median unter 0,25 pt	M
LVS10a	Geometrie-Vererbung, Stilkette, Themenfarben: Master zellweise verschmolzen (Del nimmt Geerbtes weg, Gruppen-Glieder ueber MasterShape), immer der Wert V, Stilvorlagen-Kette mit Themed, Themenfarben ueber QuickStyle, Verbinder mit <Connects>, Ecken-Rundung	M1434	M
LVS16	Gruppen bearbeiten und Groessenverhalten je Kind: proportional, fest am Anfang/Ende/mittig oder strecken, je Achse; Editor (in die Gruppe gehen, Kind ziehen/aendern/aufnehmen/herausnehmen, Gruppe ueber die Anfasser skalieren), .kdoc (groupResize), Generator/REST	M1434	M
LVS17	Ausgabe aus .kdoc: TIFF (alle Seiten), PNG/JPEG/BMP/GIF/WebP (Seite 1), EMF/WMF (vektoriell), waehlbare Aufloesung; .vsdx und .idml direkt als Arbeitsdokument statt ueber ein Zwischen-PDF	M1434	M

v4-LVS · Visio (.vsdx) — in M1429 geschlossen (6 von 6)

Neu gebaut, es gab vorher keine Zeile Visio-Code. Modul Visio, Zero-NuGet (System.IO.Compression fuers OPC/ZIP-Paket, System.Xml.Linq fuers XML, XXE-fest). Gegen zwei unabhaengige Werkzeuge belegt — LibreOffice Draw (libvisio) und die Python-Bibliothek vsdx; im Testlauf laeuft keines von beiden, die Fixtures liegen eingebettet bei.

#	Visio-Funktion	Stand bei dir	Aufw.	
LVS1	VSDX-Leser (VsdxReader)	M1429: OPC/ZIP-Paket, Seiten, Shapes, Geometrien (MoveTo/LineTo/ArcTo/Ellipse/EllipticalArcTo/CubBezTo/QuadBezTo samt den Rel...-Formen, die Bruchteile statt Zoll tragen), Text, Stilvorlagen und die Vererbung vom Master. Boegen und Ellipsen werden zu kubischen Bezierkurven aufgeloest, damit Drehung (Angle) und Spiegelung (FlipX/FlipY) auf alles gleich wirken; die Lage kommt aus Pin LocPin. Gegen LibreOffice nachgemessen auf ein Zehntel Punkt. Wirft bei keinem Eingabebyte. Gemeldet statt still: NURBSTo/PolylineTo/SplineKnot/InfiniteLine werden zur geraden Strecke genaehert (VSDX_GEOMETRIE_GENAEHERT); Bilder, Fuellmuster, Verlaeufe, Schatten, Strichmuster, Pfeilspitzen, Themen und Ebenen werden nicht ausgewertet; Text einzeilig in Helvetica ohne Umbruch und Drehung	M1429	L

#	Visio-Funktion	Stand bei dir	Aufw.	
LVS2	VSDX nach PDF (VsdxToPdf) + MCP vsdx_to_pdf	M1429: gemalt ueber die vorhandene PageCanvas-Vektor-Kette - kein zweiter Zeichenweg, wie Metafile und Svg. Die Umrechnung Zoll->Punkt geschieht an genau einer Stelle. Es gelten die Leser-Grenzen; ein Zeichen, das die Standardschriften nicht fuehren, wird weggelassen und gemeldet (VSDX_TEXT_NICHT_DARSTELLBAR) - ein Fragezeichen saehe nach Absicht aus	M1429	M
LVS3	VSDX-Schreiber (VsdxWriter) + PDF nach VSDX (PdfToVsdx) + MCP pdf_to_vsdx	M1429: das OPC-Paket mit [Content_Types].xml, den _rels, document.xml, pages.xml und je Seite ein pageN.xml. PdfToVsdx liest die Seite ueber die Anzeigeliste und den Textextraktor - wie PdfToMetafile und aus demselben Grund: PageGraphicsExtractor liefert nur umschliessende Rechtecke und gar keine Farben. Die Achse wird nicht gedreht (Visio und PDF zaehlen beide von links unten nach oben). Gemeldet statt still: Bilder, Verlaeufe, Transparenz, Beschnitt	M1429	L
LVS4	Direktimport: .vsdx als editierbares .kdoc (VsdxToKdoc) + MCP vsdx_to_kdoc	M1429: direkt, nicht ueber ein zwischengerendertes PDF - dort waere aus einem Rechteck ein Pfad aus vier Strecken geworden und aus dem Text ein Haufen Laeufe, also nichts mehr Anfassbares. Rechteck -> ShapeKind.Rectangle, Ellipse -> Ellipse, Zweipunktstrecke -> Line, sonst Polygon, mehrere geschlossene Teilpfade -> CompoundPath. .vsdx ist in DocumentConverter angeboten UND wirklich wandelbar (die Fehlerklasse IMG1/IMG22/IMG27). Dabei gefunden: Visio legt ein Rechteck ohne Close ab (der letzte LineTo geht zum Anfang zurueck) - wer nur auf den Close sieht, bekommt ein Polygon mit fuenf Punkten. Gemeldet statt still: freie Kurven werden geflacht (VSDX_KDOC_KURVE_GEFLACHT), Text einzeilig ohne Schnitt und Farbe, Gruppen loest schon der Leser auf	M1429	M
LVS5	Konnektoren mit Klebepunkten im Editor	M1429: ein Konnektor speichert seine Endpunkte nicht, sondern haelt den Namen des Start- und Ziel-Shapes samt Klebepunkt und rechnet die Stelle beim Zeichnen - nur so haelt er beim Verschieben. Dasselbe Muster, mit dem MasterPage den Namen des Masters haelt statt seiner Elemente. Klebepunkte an den Seitenmitten, Ecken und der Mitte, automatische Seitenwahl nach der Lage des Ziels, gerader und rechtwinkliger Verlauf, im .kdoc-Rundlauf gesichert	M1429	L
LVS6	Editor-Export nach .vsdx (KdocToVsdx) + MCP kdoc_to_vsdx	M1429: ebenfalls direkt aus dem Objektmodell, nicht ueber ein Zwischen-PDF. Im DocumentExporter eingehaengt, damit Editor-Knopf und kdoc_to_document dieselbe Stelle bedienen. Zwei Fehler dabei gefunden, einer nur durch das Fremdorakel: der Verbinder sass um seinen eigenen Ursprung daneben (ResolvedPoints liefert bereits absolut, X/Y wurden ein zweites Mal addiert) - die eigenen Tests prueften Art und Anmeldung, nicht die Stelle, gefunden hat es die vsdx-Bibliothek an der wirklich geschriebenen Datei; und der VsdxReader suchte die Schriftgroesse nur am Shape und in der Stilvorlage, waehrend eine echte Visio-Datei sie in <Section N='Character'> ablegt - jede Datei lieferte damit nur den Vorgabewert. Gemeldet statt still: Bilder, Tabellen, die uebrigen Fliess-Rahmen; ein Verbinder wird ein freier Streckenzug ohne Klebepunkte und Pfeilspitzen; ein zusammengesetzter Pfad verliert die Even-Odd-Fuellung; Drehung, Deckkraft, Effekte, Verlaeufe	M1429	M

v4-IMG · Bildformate — Lesen und Schreiben

Format	Lesen	Schreiben	Bemerkung
JPEG	eigener Codec		inkl. progressiv
PNG			indiziert + TrueColor
TIFF	voll	voll	Lesen: unkompr./LZW/Deflate/PackBits/CCITT G3+G4/JPEG-in-TIFF, 1–16 Bit, BigTIFF (M1332) · Schreiben: RGB/RGBA/Grau, unkompr./Deflate/PackBits/LZW/CCITT G4, mehrseitig (M1298/M1331)
GIF			LZW/Interlace/Transparenz, Median-Cut, auch animiert (M1300/M1335)
HEIF/HEIC	echte Pixel (M1147), Kachelgitter + Deckkraft + Tiefenkarte (M1339/M1340)		eigener HEVC-Intra-Decoder; schreiben -> IMG7
JPEG 2000		verlustfrei (M1306) + verlustbehaftet (M1344, ohne Ratensteuerung)	eine echte Zielgroesse braucht den PCRD-Optimierer -> IMG17
WebP	verlustfrei und verlustbehaftet (M1003)	verlustfrei (M1301) und verlustbehaftet (M1400)	animiert in beide Richtungen (M1337); die Deckkraft nur verlustfrei
JBIG2		generische Region (M1345)	Einbettung ins PDF ueber recompress_bilevel (M1346); Symbol-Woerterbuch -> IMG23
CCITT G3/G4		G4 (M1331)	Filter-Ebene; G3 wird gelesen, aber nicht geschrieben
BMP	(M1297, M1334)	(M1308, M1334)	24/32 Bit, Palettenbilder 1/4/8 Bit, BI_RLE8/BI_RLE4
TGA · Netpbm	(M1302)	(M1308)	TGA auch lauflaengen-kodiert
ICO/CUR · ANI	(M1302, M1329)	ICO/CUR (M1329)	mehrere Groessen, Klickpunkt; .ani nur lesen
ICNS (macOS)	(M1400)	(M1400)	RLE kanalweise, ARGB, eingebettetes PNG/JPEG 2000
Symbole aus .exe/.dll	(M1400)	–	PE-Ressourcenbaum -> .ico
PSD	Composite (M1304)	Composite (M1330)	kanalweise, PackBits/ZIP, Graustufen/RGB/CMYK
EMF/WMF	(M1305)	(M1328)	Vektor, in beide Richtungen — auch eine PDF-Seite als .emf
RAW/DNG	Sensordaten + Entwicklung (M1309)	x	inkl. verlustfreiem JPEG (SOF3); schreiben ist nicht vorgesehen

Format	Lesen	Schreiben	Bemerkung
JPEG XL	Behaelter, Bildkopf, Rahmenkopf und modularer Pfad - liefert Bildpunkte (M1408)	x	Schreiben -> IMG8.6; der VarDCT-Pfad (verlustbehaftet) -> IMG8.5

#	Bildformat-Funktion	Stand bei dir	Aufw.
IMG1	BMP lesen — der gemeldete Fehler ist behoben: IRasterImage.Parse erkennt die BMP-Signatur und dekodiert ueber Imaging/BmpDecoder (1/4/8/16/24/32 Bit, Palette, BI_BITFIELDS, bottom-up und top-down); Lauflaengen-Kompression BI_RLE8/BI_RLE4 in beide Richtungen und Palettenbilder schreiben (M1334)	M1297/M1308/M1334	S
IMG2	TIFF schreiben in Farbe + mehrseitig: Imaging/TiffWriter (RGB/RGBA/Graustufen, unkomprimiert, Deflate, PackBits, LZW und CCITT G4 aus M1331, mehrere Seiten ueber verkettete IFDs) — ehrlich benannte Grenzen: 8 Bit je Kanal (16 Bit und CMYK offen), ein Streifen je Seite, keine Vorhersage-Filter	M1298/M1331	M
IMG3	GIF schreiben: Imaging/GifEncoder (GIF89a, globale Farbtabelle, GIF-LZW, Transparenzindex, Median-Cut ab 256 Farben) — auch animiert (M1335: mehrere Einzelbilder, Standzeiten, Wiederholungszahl)	M1300/M1335	M
IMG4	WebP schreiben: EncodeLossless (VP8L, bit-genau inkl. Teiltransparenz), animiert (M1337) und seit M1400 auch verlustbehaftet (EncodeLossy — Einzelheiten unter IMG16)	M1301/M1337/M1400	L
IMG5	JPEG 2000 schreiben (EBCOT-Encoder; der MQ-Coder ist bereits da)	M1306 (Jpeg2000Encoder + convert_image): verlustfrei ueber die reversible 5/3-Wavelet, ein Tile/eine Schicht/LRCP. Von openjpeg als pixelgenau bestaetigt. Grenze: der verlustbehaftete 9/7-Weg braucht eine Ratensteuerung und fehlt	L
IMG9	EMF/WMF lesen -> PDF-Vektorinhalt. Praxisrelevant: Office-Dokumente tragen Grafiken oft als Metafile	M1305 (Metafile-Modul + metafile_to_pdf): Rechteck/Ellipse/Streckenzug/Vieleck/Linie/Text mit Farben und Schriftgroesse, beide Formate. Grenzen: Bitmap-Aufzeichnungen, Pfad-/Clip-Bereiche, Welt-Transformationen - gemeldet, nie still	L
IMG10	PSD lesen (zusammengesetztes Bild, ohne Ebenen-Editierung)	M1304 (PsdDecoder): kanalweise, unkompr./PackBits/ZIP, Graustufen/RGB(+Alpha)/CMYK (invertiert)/indiziert, 1/8/16 Bit; von Pillow gegengelesen. Grenzen: keine Ebenen/Masken/Effekte, kein 32-Bit-Gleitkomma	M (Nische)
IMG11	ICO / TGA / PNM (PBM,PGM,PPM) lesen	M1302 (IcoDecoder/TgaDecoder/PnmDecoder): ICO/CUR mit AND-Maske, TGA unkompr. und lauflaengen-kodiert, Netpbm P1-P6; von Pillow gegengelesen. Dabei die Erkennungs-Falle geloest, dass ein unkomprimiertes TGA wie ein Zeiger aussieht	S (Nische)

#	Bildformat-Funktion	Stand bei dir	Aufw.
IMG12	RAW/DNG lesen (Bayer-Demosaicing)	M1309 (DngReader + LosslessJpegDecoder + BayerDemosaic): Sensordaten aus dem Unterverzeichnis (nicht die Vorschau!), unkomprimiert und verlustfrei JPEG (SOF3), Dunkel-/Saettigungswert, Aufnahme-Weissabgleich, Kameramatrix, bilineares Entmosaiken. Von LibRaw gegengelesen. Grenzen: verlustbehaftetes DNG und die herstellereigenen Rohformate (CR2/NEF/ARW) fehlen; Kalibrierungs-/Vorwaertsmatrix und Kennlinien-Tabelle werden nicht ausgewertet	L (Nische)
IMG13	Werkzeug convert_image (Format->Format ueber die eigenen Codecs, mit Kompressions-/Qualitaetswahl)	M1303 (ImageConversion/ImageConverter + MCP convert_image): gelesen wird alles, was RasterImage.Parse kennt, geschrieben PNG/JPEG/TIFF (auch mehrseitig)/GIF/WebP/BMP/TGA/PNM/ICO/PSD/JPEG 2000; optionales Verkleinern ueber MaxEdge. Die Wandlung liegt in einem eigenen Kompositionsmodul, weil Imaging das Modul Content nicht importieren darf	S
IMG14	EXIF lesen und schreiben	M1400 (Imaging/ExifCodec auf dem vorhandenen TiffDirectory, kein zweiter IFD-Leser; ImageMetadataCodec als Fassade): Aufnahmezeit, Kamera/Objektiv, Belichtungszeit, Blende, ISO, Brennweite, Ausrichtung und GPS in JPEG-APP1, PNG-eXIf und im IFD0 einer TIFF. Fixtures aus Pillow. Ueber MCP image_metadata/set_image_metadata. Grenzen: TIFF nur lesend (die Bilddaten unangetastet zu lassen waere ein eigener Schritt); nur eine praxisnahe Tag-Teilmenge; das Anwenden der Ausrichtung (Drehen) ist nicht Teil des Codecs	M
IMG15	IPTC/XMP im Bild lesen und schreiben	M1400 (Imaging/IptcCodec fuer die IIM-Datensaetze in der Photoshop-8BIM-Ressource, XmpImagePacket als XXE-fester RDF-Leser/-Bauer): Titel, Beschreibung, Urheber, Rechte, Stichwoerter in JPEG (APP1/APP13), PNG (iTXt), TIFF (Tag 700, lesend) und WebP (XMP -Block inkl. VP8X-Erzeugung). Geschrieben wird IPTC UND XMP aus derselben Quelle - aeltere Werkzeuge lesen nur das eine, neuere nur das andere. Grenzen: ein JPEG mit vorhandener Photoshop-Ressourcensammlung verliert deren uebrige Eintraege; komprimiertes iTXt wird uebergangen	M
IMG16	WebP verlustbehaftet schreiben (VP8-Intra-Encoder)	M1400 (WebPEncoder.EncodeLossy + Imaging/Vp8Encoder): boolescher Entropie-Codierer mit Uebertrags-Fortpflanzung, Rahmenkopf, Intra-Vorhersage (16x16 DC/V/H/TM, alle zehn 4x4-Modi, 8x8-Chroma), 4x4-DCT + Walsh-Hadamard, Quantisierung, Token-Kodierung - vorhergesagt wird gegen die rekonstruierten Nachbarn. Geprueft ueber den eigenen Vp8LossyDecoder und libwebp (Guete 90: 1,05 mittlere Abweichung je Kanal). Grenzen: Schleifenfilter aus, keine Segmentierung, eine Token-Partition, keine Wahrscheinlichkeits-Anpassung, keine Ratensteuerung, keine Deckkraft (kein ALPH-Block)	XL
IMG18	BMP, TGA und Netpbm schreiben	M1308 (BmpEncoder/TgaEncoder/PnmEncoder + convert_image): BMP 24/32 Bit bottom-up mit Zeilenauffuellung, TGA unkomprimiert und lauflaengen-kodiert, Netpbm P4/P5/P6. Von Pillow als pixelgenau bestaetigt. Grenzen: keine Palettenbilder, keine BMP-RLE, Netpbm nur binaer	S

#	Bildformat-Funktion	Stand bei dir	Aufw.
IMG19	ICO/CUR schreiben	M1329 (IcoEncoder + ImageFormat.Ico): mehrere Groessen in EINER Datei, PNG- und DIB-Eintraege mit AND-Maske, von Pillow pixelgenau gegengelesen. Grenzen: keine Palettenbilder, Kante hoechstens 256	M
IMG20	PSD schreiben (Composite)	M1330 (PsdEncoder + ImageFormat.Psd): kanalweise, unkomprimiert/PackBits/ZIP, Graustufen/RGB(+Alpha)/CMYK invertiert. Von Pillow pixelgenau gegengelesen (ausser ZIP, das Pillow im Composite-Abschnitt nicht kann). Dabei ein stiller Leser-Fehler behoben: der ZIP-Zweig las jeden Kanal ab derselben Stelle. Grenzen: keine Ebenen, 8 Bit, kein PSB, kein ICC	M
IMG21	EMF/WMF schreiben — die Vektor-Gegenrichtung zu M1305	M1328 (MetafileWriter + PdfToMetafile + pdf_to_metafile): beide Formate aus dem Zeichenmodell, und eine PDF-Seite ueber die Anzeigeliste (echte Pfade mit aufgeloesten Farben, nicht die blossen Rechtecke des Grafik-Extraktors). Dabei zwei stille Leser-Fehler behoben: Pinsel und Stift faerbten einander, und die Strichstaerke ging verloren. Grenzen: keine Bilder, kein Beschnitt/Verlauf/Transparenz, Bezierkurven werden in Strecken zerlegt	M
IMG24	Mauszeiger vollstaendig (.cur/.ani)	M1329 (IcoDecoder.DecodeAll/IsCursor + AniDecoder): Klickpunkt lesen und schreiben, .ani mit allen Einzelbildern und ihren Standzeiten aus dem rate-Block; .ani auch im Erkennungsweg. Grenze: .ani wird nicht geschrieben, rohe DIB-Bloecke (bfAttributes Bit 1) werden uebersprungen statt geraten	S–M
IMG25	macOS-Symbole (.icns) lesen und schreiben	M1400 (IcnsDecoder/IcnsEncoder): is32/il32/ih32/it32 (RGB, roh pixel-verschachtelt oder Apple-RLE kanalweise - ein am fremden Leser gemessener Unterschied, den ein diskriminierender Test faengt) samt Maskenbloecken, ic04/ic05 (ARGB) und ic07-ic14 (eingebettetes PNG/JPEG 2000 ueber die vorhandenen Decoder). In RasterImage.Parse, ImageConverter und convert_image erreichbar. Grenzen: nur die sieben Standardgroessen, nur quadratisch	M
IMG26	Symbole aus Programmen holen (.exe/.dll)	M1400 (Imaging/PeIconExtractor + MCP extract_exe_icon): PE32/PE32+-Ressourcenbaum, RT_GROUP_ICON/RT_ICON ueber den vorhandenen IcoEncoder zu einer .ico zusammengesetzt - samt der zwei Fallen (der Gruppeneintrag traegt eine Ressourcen-ID statt eines Versatzes, und die Versaetze sind RVAs). Real geprueft an notepad.exe und explorer.exe. Grenze: auf neueren Windows-Fassungen liegen die Ressourcen mancher Systemdateien ausgelagert (.mun) und fehlen im Hauptmodul - bei shell32.dll gemessen	M

#	Was fehlt	Warum es vorkommt	Aufw.
IMG27	TIFF: LZW und CCITT G4 schreiben; BigTIFF lesen. Berichtigung: gekachelte TIFFs (TileWidth/TileLength) und die planare Ablage (PlanarConfiguration 2) werden seit M122 bereits gelesen — das stand hier faelschlich als offen	M1331 (LzwEncoder + CcittFaxEncoder + TiffCompression.Lzw + TiffBilevelCompression.Group4): beide Codierer von Pillow pixelgleich gegengelesen; der LZW-Codierer dient zugleich PDFs /LZWDecode. Grenzen: kein Praediktor, nur G4 (nicht G3). M1332: BigTIFF (Version 43) wird gelesen - 8-Byte-Versaetze, 20-Byte-Eintraege, die Typen LONG8/SLONG8/IFD8, auch im Erkennungsweg; Fixture aus einem fremden Schreiber (Pillow). Grenze: die API nimmt ReadOnlyMemory<byte>, also passt eine Datei ueber 2 GB gar nicht erst hinein - BigTIFF heisst hier „BigTIFF-Struktur", nicht „Datei ueber 4 GB"	M–L
IMG28	PNG: Adam7-Interlace und 16 Bit je Kanal schreiben; dazu APNG lesen und schreiben	M1333 (PngEncoder.WriteTrueColor16 + interlace-Schalter): 16 Bit in RGB und Graustufen mit und ohne Deckkraft, Adam7 in allen sieben Durchgaengen auch bei ungeraden Massen. Von Pillow bestaetigt - das 16-Bit-Graustufenbild kommt als I;16 mit exakt den geschriebenen Werten zurueck, das verschachtelte pixelgleich zum glatten. M1336: APNG lesen und schreiben (siehe IMG30)	M
IMG30	GIF und WebP: Animationen — mehrere Einzelbilder lesen und schreiben	M1335 (GIF, GifDecoder.DecodeAnimation + GifEncoder.EncodeAnimation): alle Einzelbilder als fertig zusammengesetzte Vollbilder (Ausschnitt, Entsorgungsart 0/1/2/3, Transparenz), Standzeiten und Wiederholungszahl in beide Richtungen; von Pillow in beide Richtungen bestaetigt. Grenze: der Schreiber optimiert nicht auf Ausschnitte. M1336: APNG (Content/ApngImage) in beide Richtungen - acTL/fcTL/fdAT, Ausschnitt an seiner Stelle, Entsorgungs- und Mischart, Standzeit als Bruch, echte Deckkraft; ein geschriebenes APNG bleibt ein gueltiges PNG, das das erste Einzelbild zeigt. Von Pillow in beide Richtungen bestaetigt. M1337: WebP-Animation (WebPDecoder.DecodeAnimation + WebPEncoder.EncodeAnimation) - VP8X/ANIM/ANMF mit eingeschachteltem Bildblock, Stelle in Zweierschritten, Masse um eins vermindert, Misch- und Entsorgungsbit; von Pillow in beide Richtungen bestaetigt. Grenze: geschrieben wird nur verlustfrei (der VP8-Codierer ist IMG16)	M
IMG31	BMP: Lauflaengen-Kompression und Palettenbilder	M1334 (BmpEncodeOptions + RLE im BmpDecoder): BI_RLE8/BI_RLE4 in beide Richtungen, Palettenbilder mit 1/4/8 Bit. Von Pillow pixelgleich gegengelesen. Zwei Befunde: mein spezifikationskonformes RLE4 mit ungerader Absolutlauf-Laenge wird von verbreiteten Lesern falsch verstanden (sie ruecken um Bytes statt Bildpunkte vor) - deshalb schreibe ich nur gerade Laengen; und die Farben werden nicht quantisiert, zu viele Farben werden benannt abgelehnt	S
IMG32 11/11	JPEG 2000: das volle Codestream-Profil (Kacheln, Schichten, Precincts, Behaelter-Boxen)	Untertabelle unten: IMG32.1–32.11 — seit M1408 vollstaendig	L
IMG33	HEIF: mehrteilige Bilder (grid), Deckkraft-Ebene und Tiefenkarten lesen	M1339 (grid: Kachelgitter zusammengesetzt - Zeilen/Spalten minus eins, Kachelreihenfolge aus iref/dimg) + M1340 (Deckkraft-Ebene und Tiefenkarte ueber iref/auxl + die auxC-URN; die Deckkraft wird angewandt, die Tiefenkarte als DepthMap geliefert). Benannte Grenze: bei einem Kachelgitter wird die Deckkraft nicht angewandt	L

Die grossen Bildformat-Punkte im Einzelnen

#	Scheibe	Stand
IMG6.1	OBU-Schicht, Sequenzkopf, av1C-Box	M1310 - von libavif gegengelesen
IMG6.2	Rahmenkopf (Einzelbild-Pfad)	M1400
IMG6.3	Arithmetischer Symbol-Decoder + Vorgabe-CDF-Tabellen	M1400/M1401
IMG6.4	Kachelgruppe und Partitionsbaum	M1400
IMG6.5	Intra-Vorhersage (alle 13 Modi) + Dequantisierung	M1400
IMG6.6	Inverse Transformationen (DCT/ADST/Identitaet/WHT, 2D)	M1402 - gegen die geschlossene Kosinus-/Sinus-Form geprueft, 0,006 % Abweichung
IMG6.7	Koeffizienten-Auslesung (§5.11.39) und Zusammenbau je Block	M1408 - Abtastreihenfolgen, Vorgabe-Wahrscheinlichkeiten und der Koeffizienten-Leser; die Bildpunkte haengen jetzt allein an IMG6.8

#	Scheibe	Stand
IMG7.1	HEIF-Behaelter (ftyp/meta/iinf/iloc/iprp/ipco/ipma/mdat)	M1400
IMG7.2	CABAC-Codierer	M1400
IMG7.3	Vorwaerts-Transformation und Quantisierung	M1400
IMG7.4	Parametersaetze VPS/SPS/PPS + Slice-Kopf, Intra-Encoder	M1400 - heif-convert (libheif) liest den Strom
IMG7.5	Deckkraft als Hilfsbild (auxl)	M1400
IMG7.6	Sammlungen und Kachelgitter (grid) schreiben	M1400

#	Scheibe	Stand
IMG8.1	Behaelter (Codestream FF 0A und ISO-BMFF), Bitleser, Bildkopf	M1400 - Feldreihenfolge aus der libjxl-Quelle 0.12.0, 17 Pruefdateien von libjxl erzeugt
IMG8.2	Rahmenkopf und TOC	M1408 - dabei fehlten im Bildkopf zwei Felder, ohne die der Rahmenkopf dahinter nicht an seiner Stelle begann; der Bitleser wurde vom ref struct zur Klasse, damit Kopf- und Entropieschicht denselben Leser teilen statt zweier Fassungen derselben Bitarithmetik
IMG8.3	Entropie-Decoder (Praefix-Codes + rANS, Hybrid-Ganzzahlen, Kontext-Abbildungen)	M1408 - wie angekuendigt zusammen mit IMG8.4 belegt: libjxl exportiert keine Zwischenzustaende, also ist der Beleg das fertige Bild
IMG8.4	Modularer Pfad (MA-Baeume, Selbstkorrektur-Praediktion, Squeeze/RCT)	M1408 - damit liefert JPEG XL Bildpunkte: .jxl steht jetzt in der Tabelle des Erreichbarkeits-Waechters statt in dessen Ausnahmeliste, kommt also ueber RasterImage.Parse UND DocumentConverter an

#	Scheibe	Stand
IMG23.1	Generische Region, arithmetisch (Seiteninfo, Region, Seitenende)	M1345 - kleiner als der eigene CCITT-G4-Schreiber
IMG23.2	Einbettung ins PDF (PdfBilevelRecompressor + recompress_bilevel)	M1346 - dabei zwei Fehler gefunden: das /Decode gehoert NICHT dazu, und der eigene Leseweg ueberging es
IMG23.3	Symbol-Woerterbuch + Textregion + Restbild-Region	M1400 - Textseite auf ein Drittel (190 statt 511 Bytes)
IMG23.4	Fremdverifikation und Freigabe	M1402 - zwei Ursachen gefunden (SBSYMCODELEN ist bei EINER Form 0, nicht max(1,...); die Endmarke FF AC fehlte); gegen MuPDF/jbig2dec und PDFium je 0 abweichende Bildpunkte, Vorgabe wieder AN
IMG23.7	MMR-Kodierung als Alternative zur arithmetischen	M1408 - ueber den vorhandenen Filters/CcittFaxEncoder, kein zweiter Codierer
IMG23.8	JBIG2Globals — ein Woerterbuch fuer mehrere Seiten	M1408 - der eigentliche Gewinn bei einem mehrseitigen Scan: das Symbolwoerterbuch steht einmal im Globals-Strom statt in jeder Seite

#	Scheibe	Stand
IMG29.1	12 Bit je Komponente lesen (SOF1)	M1338 - Fixture von libjpeg-turbo; Ausgabe auf 8 Bit heruntergerechnet (benannte Grenze)
IMG29.2	Progressiv schreiben (SOF2, Spektralauswahl)	M1343 - libjpeg liest es als progressiv und pixelgleich
IMG29.3	Arithmetische Kodierung lesen, sequenziell (SOF9)	M1400 - erst das Fremdorakel zeigte drei Fehler (Abweichung 224-255 je Kanal, obwohl alle eigenen Tests gruen waren)
IMG29.4	SOF10 — progressiv arithmetisch	M1408
IMG29.5	Restart-Intervalle in arithmetischen Scans	M1408

#	Scheibe	Stand
IMG32.1	JP2-Behaelter schreiben (Signaturbox/ftyp/jp2h/jp2c)	M1341 - dabei ein stiller Fehler behoben: .jp2 bekam den rohen Codestream
IMG32.2	Kacheln lesen und schreiben	M1342 - je Kachel eigene Wavelet und Teilbaender, Randkacheln kleiner
IMG32.3	Ratensteuerung (PCRD-opt, echte Zielgroesse in Bytes)	M1400
IMG32.4	Mehrere Qualitaetsschichten	M1400
IMG32.5	Mehrere Teilstuecke je Kachel (Tile-Parts)	M1400
IMG32.6	Behaelter-Boxen res /colr+ICC/cdef samt Deckkraft	M1400
IMG32.7	Kleine Precincts lesen und schreiben	M1400 - dabei ein stiller Lesefehler geschlossen: eine fremde Datei mit Precincts kam falsch heraus (Bildpunkt 0 statt 17), ohne dass ein einziger eigener Test rot wurde
IMG32.8	Mehrkomponenten-Transformation (RCT reversibel, ICT irreversibel)	M1405

#	Scheibe	Stand
IMG32.9	Andere Progressionsverlaeufe als LRCP (RLCP, RPCL, PCRL, CPRL)	M1406 - alle fuenf, lesen und schreiben
IMG32.10	Bittiefen ausser 8 (1–16 Bit, mit und ohne Vorzeichen)	M1407 - 12 und 16 Bit las der Dekoder laengst richtig (die Zeile war zu pessimistisch, gegen zwei opj_compress-Dateien belegt). Der Fund lag bei 38 Bit, die der Standard zulaesst: 1 << 37 maskiert C# stillschweigend auf 1 << 5 - die Datei wurde weder abgelehnt noch richtig gelesen. Ueber 31 Bit wird jetzt benannt abgelehnt. Schreibseite 1..8 (mehr hiesse, Abstufungen zu erfinden, die RgbaImage nicht traegt)
IMG32.11	Ein Precinct-Mass je Aufloesungsstufe beim Schreiben; Precinct-Raster am Bild- statt am Kachelursprung	M1408

v4-FNT · Schriften — alle Arten, beide Richtungen

Art	Lesen/Rendern	Einbetten/Subsetten	Als Datei ausgeben
TrueType (glyf)			.ttf byte-treu
OpenType/CFF			.otf
bare CFF (Type1C/CIDFontType0C)			-> OTF gewrappt
CID-keyed CFF		subsettet (M291)	
CFF2 (variabel)		instanziert -> statisch	
Type1 (PFB/eexec)			.pfb/.pfa
Type3 (Prozedur-Glyphen)		erzeugbar	— (kein Dateiformat)
Type42 (PostScript)			— (kein Dateiformat)
Variable Fonts (fvar/gvar/avar/HVAR)		instanziert	
Farbfonts COLR/CPAL v0+v1			
WOFF / WOFF2		(M1311)	Container-Umpacken in jeder Richtung
Farbfonts CBDT/EBDT/sbix (Bitmap-Emoji)	M1323	nur sbix-PNG	lesen+zeichnen, schreiben offen
Farbfonts SVG-in-OpenType	M1326	(ColorFontSvgEncoder, FED13-Rest)	lesen+zeichnen (als Vektor, nicht gerastert)
TrueType-Hinting (fpgm/prep-Bytecode)	(opt-in im Rasterizer)	—	—
Multiple Master · BDF/PCF/.fon/.fnt	M1316/M1347	BDF+PSF (PCF/FON offen)	

#	Schrift-Funktion	Stand bei dir	Aufw.
FNT1	WOFF + WOFF2 schreiben (Encoder; Brotli/zlib sind in der BCL)	M1311 (WoffEncoder/Woff2Encoder): sfnt -> WOFF (zlib je Tabelle) und -> WOFF2 (ein Brotli-Strom), mit korrekter urspruenglicher Groesse; Rundlauf ueber die eigenen Entpacker belegt. Grenze: kein WOFF2-Transform der glyf/loca-Tabellen (die Datei ist dadurch groesser als eine von fontTools, aber gueltig)	M
FNT2	Bitmap-Farbfonts CBDT/CBLC + sbix lesen und zeichnen	M1323 (Fonts/ColorBitmapFont, PageCanvas.DrawBitmapColorText) - sbix nur PNG	M
FNT3	SVG-in-OpenType (SVG -Tabelle) rendern	M1326 (Fonts/ColorFontSvg, PageCanvas.DrawSvgColorText)	M
FNT4	TrueType-Hinting (Bytecode-Interpreter fpgm/prep/glyf-Instruktionen) fuer scharfe kleine Groessen im Rasterizer	Fonts/TrueTypeHinting: nahezu der volle Befehlssatz (Stack/Arithmetik/Logik/Kontrollfluss IF/ELSE/EIF/JMPR/JROT/JROF, beide Zonen - Glyph und Twilight -, alle Rundungszustaende inkl. SROUND/S45ROUND, MDAP/MIAP/MDRP/MIRP/SHP/SHC/SHZ/SHPIX/ALIGNRP/ISECT/IUP, DELTAP1-3/DELTAC1-3, WS/RS, FDEF/LOOPCALL/CALL). Eine wirklich nicht umgesetzte Instruktion (SDPVTL/IP/MSIRP/ALIGNPTS/FLIPPT/WCVTF/IDEF/GPV/GFV - die verbleibende, benannte v2-Luecke) bricht weiter benannt ab, statt zu raten. Angebunden an GlyphRasterizer.Rasterize (Parameter hinting, Vorgabe aus - ohne ihn byte-identisch); wirkt auf die bereits geglaetteten Umriss-Punkte (ehrlich benannte Naeherung, kein Zugriff auf die rohen glyf-Kontrollpunkte), schlaegt ein Programm fehl, faellt der Rasterizer auf den ungehinteten Umriss zurueck statt zu werfen	L
FNT5	Werkzeug convert_font - Schriftdatei in alle Richtungen wandeln (ttfotfwoffwoff2pfb/pfa) + Pruefstand-Seite	M1311 (Fonts/FontConverter + MCP convert_font): Container umpacken in jeder Richtung, bare-CFF in einen OpenType-Rahmen, pfb pfa. Grenze: es werden KEINE Umrisse umgerechnet - glyf CFF und Type1 sfnt werden benannt abgelehnt statt kaputt geschrieben; kein Subsetting. Pruefstand-Seite fehlt noch	S
FNT6	Type1 Multiple Master (Instanzieren wie bei Variable Fonts)	Fonts/Type1MultipleMaster: aus einer Achsposition wird ein fester Schnitt gerechnet	M (Nische)
FNT7	Alt-Bitmapschriften BDF/PCF/.fon/.fnt lesen	M1316 (Fonts/BitmapFont: BDF, PCF, Windows .fnt/.fon - FNT-Fassung 2.0/3.0 + NE-Behaelter, mehrere Groessen je Datei; gegen 192 reale .fon-Dateien gehalten). Diese Zeile stand faelschlich auf „fehlt" - nachgezogen (M1316 hatte sie gebaut, aber nicht hier vermerkt)	S (Nische)

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

Format	Wofuer	Lesen	Schreiben
dfont / Suitcase	macOS	M1452: sfnt-Ressourcen (DfontReader, M1364) und die klassischen Bitmap-Ressourcen FOND/NFNT (MacBitmapFont)	M1452: sfnt (DfontWriter, FED16) und FOND/NFNT (MacBitmapFontWriter)
TTF (sfnt/glyf)	Web, Druck, System		(SfntWriter, FontBuilder - FED2)

Format	Wofuer	Lesen	Schreiben
OTF (sfnt/CFF)	Web, Druck		(CffOpenTypeWrapper, FontBuilder - FED2)
WOFF	Web		M1311 (WoffEncoder)
WOFF2	Web		M1311 (Woff2Encoder)
Variable Fonts (fvar/gvar/HVAR)	Web, Druck	inkl. Instanzieren	FED9 (VariableFontBuilder - fvar/gvar/avar/STAT; kein HVAR, die Breiten variieren ueber die Phantompunkte in gvar)
CFF2	Druck		(Cff2Writer - VarStore/blend aus statischen CFF-Mastern, die CFF-Entsprechung zu FED9s glyf/gvar)
CID-keyed CFF (ROS)	CJK-Druck		FED10 (CidKeyedCffWriter - Identitaets-Charset CID=GID, ein FDArray-Eintrag)
Type1 (PFB/PFA)	Alter Druck		(Type1FontConverter)
Type3	PDF-intern		(PDF-Erzeugung)
Type42	PostScript		FED11 (Type42Writer, strukturell geprueft - kein Ghostscript auf dieser Maschine)
TTC/OTC (Sammlungen)	System	FED12 (FontCollection.Read)	FED12 (FontCollection.Write, Tabellen-Dedup)
COLR/CPAL (Farbe)	Web, System	v0+v1	v0 (ColorFontEncoder, M1364) + v1 (ColorFontV1Encoder, FED13-Rest)
CBDT/EBDT · sbix (Bitmap-Emoji)	System	(M1323)	(ColorBitmapFontEncoder - CBDT-Bildformat 17 bzw. sbix-Tag "png ", je genau eine Aufloesungsstufe; Bildformate 18/19, monochrome Formate und "dupe"/"jpg "/"tiff"/"pdf " bei sbix bleiben aussen vor)
SVG-in-OpenType	Web	(M1326)	(ColorFontSvgEncoder, FED13-Rest)
EOT	Web (alt, IE)		
SVG-Fonts	Web (abgeloest)		
BDF · PCF · PSF	Linux/Konsole	(M1316 BDF/PCF, M1364 PSF1/PSF2)	FED15 (BitmapFontWriter.WriteBdf/WritePsf1/WritePsf2/WritePcf)
FON/FNT	Windows (alt)	(M1316)	FED15 (BitmapFontWriter.WriteFnt - das blanke .fnt; der NE-Behaelter .fon bleibt ein reiner Leseweg)
UFO · designspace	Quellformat der Editoren		
SFD (FontForge) · TTX (fonttools-XML)	Austausch	SFD; TTX (Metrik, Kontur, zusammengesetzte Glyphen, CFF , post)	SFD; TTX (dieselben)

Format	Wofuer	Lesen	Schreiben
AFM/PFM (Metriken)	Alter Druck		

#	Funktion	Stand bei dir	Aufw.
FED1	Glyphen-Umrisse aendern - Punkte/Kurven lesen, bearbeiten und zurueckschreiben (glyf und CFF-Charstrings) - M1364 (Fonts/GlyphPath = bearbeitbares Modell mit exakter Grad-Erhoehung quadratisch->kubisch und angenaeherter Gegenrichtung; Fonts/GlyphOutlineEditor liest/schreibt beide Umriss-Arten; Fonts/GlyphOutlineTables = glyf/loca + Type-2-Charstrings + CFF-Tabelle). Grenzen: kubisch->quadratisch ist eine Annaeherung bis zur Fehlerschranke (Vorgabe 0,5 Einheiten); ein ERSETZTER Glyph verliert Hinting und wird einfach statt zusammengesetzt (unveraenderte behalten ihre Bytes); bei CFF wird die ganze Tabelle neu gebaut, also entfallen Subroutinen, Hints und die Glyphnamen - eine CID-keyed CFF wird benannt abgelehnt		L
FED2	Neue Schrift von Grund auf: Glyphen aus Umrissen, Metriken, cmap, Namen, OS/2 - als .ttf/.otf schreiben - M1364 (Fonts/FontBuilder.BuildTrueType/BuildOpenTypeCff; .notdef wird selbsttaetig ergaenzt, die lsb folgt der echten xMin). Grenzen: cmap nur Format 4 (BMP - Format 12 ist FED5), post 3.0 ohne Glyphnamen; kein Kerning (FED4), keine GSUB/GPOS, kein Hinting, keine zusammengesetzten Glyphen, keine Variationsachsen (FED9)		L
FED3	Wandeln in jede Richtung - TrueType CFF (quadratisch kubisch umrechnen), Type1 -> OpenType, sfnt -> WOFF/WOFF2, Type42 -> sfnt (ueber das neue Kompositionsmodul FontConversion/Type42SourceReader - Fonts darf PostScript nicht importieren, dieselbe Loesung wie ImageConversion in M1305) und CFF2 -> CFF		L
FED4	Metriken und Kerning bearbeiten: Fonts/FontMetricsEditor.SetMetrics (Vorschubbreiten + linke Seitenraender je Glyphe, hmtx/hhea) und SetKerningPairs (klassische kern-Tabelle, Format 0). Grenze: geschrieben wird kern, nicht GPOS - gelesen werden beide (GlyphKerning)		M
FED5	Zeichenzuordnung bearbeiten: Fonts/FontCmapEditor.SetCmap (cmap anlegen/aendern, Format 4/6/12) und Fonts/FontPostNames.SetNames/Read (Glyphennamen, post Format 2.0)		M
FED6	Zusammenfuehren/Aufteilen: Fonts/FontGlyphSetEditor.Merge (Glyphen aus mehreren TrueType-Schriften in eine, Composite-Referenzen umgeschrieben) und ExtractSubset (Teilmenge herausloesen mit Umbenennen ueber FontPostNames und Codepunkt-Neuzuordnung ueber FontCmapEditor - mehr als das reine TrueTypeSubsetter-Subsetting). Grenze: nur glyf/loca-Schriften, gleiches UnitsPerEm, Layout-Tabellen werden beim Merge verworfen		M

#	Funktion	Stand bei dir	Aufw.
FED7	Schrift-Editor im Pruefstand: Glyphe waehlen, Umriss sehen und ziehen, Metriken setzen, Vorschau, Speichern - die Oberflaeche ueber FED1-FED6 - (/schrifteditor, Playground/Model/FontEditorSupport + Components/Pages/FontEditor.razor): Umriss-Punkte live per Zeiger ziehen (reine SVG-Pointer-Ereignisse auf dem SVG-Wurzelelement, keine JS-Bruecke noetig), Vorschubbreite setzen, Struktur-Befund (FED8) anzeigen, mehrere Glyphen aendern und in einem Zug speichern (GlyphOutlineEditor.SetOutlines + FontMetricsEditor.SetMetrics)		L
FED8	Pruefer fuer selbst gebaute Schriften: Fonts/FontStructureChecker.Check - Pflichttabellen, Tabellen-Pruefsummen, Metrik-Widersprueche (numberOfHMetrics/hmtx-Laenge, loca/glyf-Konsistenz, cmap-GID-Bereich). Das Gegenstueck zu Extraction.FontHealthChecker (der prueft Schriften in einem fremden PDF, dieser eine eigenstaendige Schriftdatei)		M
FED9	Variable Fonts SCHREIBEN: aus mehreren fertigen statischen Schnitten (z. B. Regular+Bold) eine variable Schrift bauen - M1384 (Fonts/VariableFontBuilder.Build + GvarWriter): fvar (Achsen + benannte Instanzen), gvar (Punkt-Deltas mit eingebettetem Peak-Tupel und aneinandergrenzenden Wirkungszonen bei mehreren Mastern auf einer Achse), avar (Umbiegung der Skala), schlanke STAT; die Achs- und Instanznamen wandern in die name-Tabelle. Nicht interpolierbare Glyphen (andere Punktzahl, andere Konturgliederung, andere Komponenten) werden benannt abgelehnt. Fremdverifiziert mit fontTools (einmalig beim Bauen, kein Python im Testlauf). Grenzen: kein IUP (die Deltas stehen fuer alle Punkte - normgerecht, nur groesser); keine Eck-Master (ein Schnitt darf hoechstens eine Achse vom Standard abweichen - das volle Variationsmodell mit sich ueberlagernden Zonen fehlt); kein HVAR/MVAR/VVAR/cvar (die Vorschubbreiten variieren ueber die Phantompunkte in gvar); nur TrueType-Umrisse (CFF2 schreiben waere ein eigener Weg); STAT ohne AxisValue-Eintraege		XL
FED10	CID-keyed CFF schreiben (ROS, FDArray/FDSelect) — die Form, in der CJK-Schriften ausgeliefert werden		L - (Fonts/CidKeyedCffWriter.MakeCidKeyed): aus einer name-keyed bare-CFF wird eine CID-keyed bare-CFF (ROS + Identitaets-Charset CID=GID + ein FDArray-Font-DICT + FDSelect Format 3 mit einem Bereich); CharStrings, Global-/lokale Subroutinen und das Private-DICT werden byteidentisch uebernommen. Ueber CffOpenTypeWrapper.Wrap in eine ladbare .otf verpackbar. Grenze: genau ein Font-DICT (keine Aufteilung auf mehrere Private-DICTs je Glyphengruppe); nur CharstringType 2; lokale Subrs muessen unmittelbar hinter dem Private-DICT liegen.

#	Funktion	Stand bei dir	Aufw.
FED11	Type42 schreiben (sfnt in PostScript verpackt) — fuer die PostScript-Ausgabe		S - (Fonts/Type42Writer.Write): /sfnts traegt die vollen sfnt-Rohbytes hexadezimal (in 65534-Byte-Bloecke zerlegt), /CharStrings bildet jeden Glyphnamen auf seine GID ab (der native Type-42-Mechanismus des Interpreters uebernimmt die Umriss-Interpretation). Grenze: Glyphnamen gNN (keine post-Namen-Aufloesung), /Encoding StandardEncoding; nicht gegen Ghostscript verifiziert (auf dieser Maschine nicht installiert) - strukturell geprueft ueber den Rundlauf der eingebetteten sfnt-Bytes.
FED12	Sammlungen (TTC/OTC) lesen und schreiben: mehrere Schriften mit geteilten Tabellen in einer Datei		M - (Fonts/FontCollection): Read zerlegt jede Schrift der Sammlung in ihre Tabellen, Write setzt mehrere Schriften wieder zusammen und legt byteidentische Tabellen nur einmal ab (Dedup - der Sinn des Formats). Mit fontTools (TTCollection) als unabhaengigem Orakel gegengelesen. Grenze: TTC-Kopfversion 1.0 (kein DSIG-Verzeichnis wie Version 2.0); head.checkSumAdjustment wird nicht neu gerechnet (bei geteilten Tabellen ueber mehrere Schriften nicht eindeutig auf eine Schrift beziehbar, von Betrachtern praxisnah nicht geprueft).
FED13	Farbfonts schreiben: COLR/CPAL v0 (M1364, ColorFontEncoder) - Rest : COLR v1 (Fonts/ColorFontV1Encoder - Verlaeufe linear/radial/konisch ueber die allgemeine PaintTransform-Affine, Komposition ueber PaintComposite, PaintColrLayers+LayerList wie bei v0; gemeinsamer Tabellen-Zusammenbau ColorFontEncoder.ReplaceTables mit v0) und die SVG -Tabelle (Fonts/ColorFontSvgEncoder - sortierte, ueberlappungsfreie Dokumentliste, optionales Gzip, ueberlappende Bereiche werden benannt abgelehnt statt eine fuer die Binaersuche des Lesers unbrauchbare Datei zu schreiben). Grenze: die variablen v1-Paint-Formate (M359) werden nicht geschrieben (jede Schrift ist eine statische Instanz)		L
FED14	Web-Altlasten: EOT und SVG-Fonts lesen und schreiben — beide abgeloest, aber in Bestaenden vorhanden. Fonts/EotFont+Fonts/SvgFontFile, je Leser/Schreiber. EOT: Version 0x00020001, unkomprimiert (kein MicroType-Express-Codec — benannt abgelehnt); die eingebettete sfnt-Schrift bleibt ueber den vorhandenen TrueTypeFont-Leser gueltig. SVG-Fonts: das abgeloeste <font>-Element mit <glyph>-Pfaddaten (eigener minimaler d-freier Attributweg, kein Svg-Modul-Import — Fonts darf Svg nicht importieren)		M
FED15	Bitmapschriften schreiben: BDF, PSF1/PSF2, PCF und Windows FON/FNT (Fonts/BitmapFontWriter: WriteBdf/WritePsf/WritePcf/WriteFnt) — je gegen den vorhandenen Leser gerundlauft; die zwei Format-Fallen sind getestet: PCF legt die Bits je nach Kopf LSBit zuerst ab, das Windows-FNT die Punktmuster spaltenweise		M
FED16	macOS-Bestaende: dfont/Suitcase lesen (Fonts/DfontReader) und schreiben (Fonts/DfontWriter) — sfnt-Ressourcen in der klassischen Resource-Fork, mit den 24-Bit-Versaetzen. Ehrlich benannte Grenze: nur sfnt-Ressourcen, keine FOND/NFNT (die alten Bitmap-Suitcases)		M

#	Funktion	Stand bei dir	Aufw.
FED17	UFO + designspace lesen und schreiben — das Quellformat, in dem Schriften entworfen werden; damit laesst sich mit anderen Editoren zusammenarbeiten. Fonts/UfoFont+Fonts/UfoSerializer (reine Bytes, wie KdocSerializer) + Fonts/UfoFolder (die einzige Stelle mit Dateisystem-Zugriff, wie KdocFolder) + Fonts/DesignSpaceFile. v1-Grenze: die gaengige fontinfo.plist-Teilmenge (nicht alle 200+ moeglichen Schluessel), kein lib.plist/Kerning/Gruppen/Komponenten-Glyphen		M
FED18	Austauschformate: FontForge-SFD und fonttools-TTX (XML) lesen und schreiben. Fonts/SfdFont + Fonts/TtxFont. Umgesetzt: Kopfmetriken, Glyphenreihenfolge/-namen, Fortschrittsbreiten, Unicode-Zuordnung (cmap format 4), bei SFD die vollen Kontur-Umrisse (SplineSet), bei TTX die Umriss-Geometrie einfacher UND (Rest-Teil 2) zusammengesetzter glyf-Glyphen (TtxFont.Glyphs/TtxGlyphComponent, baut auf Fonts/GlyphPath auf, M1364 - kein zweiter Umriss-Weg; die implizite Mittelpunkt-Regel bei zwei aufeinanderfolgenden Kontrollpunkten ist umgesetzt und ueber eine hand-gebaute Fixture belegt; eine zusammengesetzte Glyphe traegt jetzt ihre Verweis-Bestandteile samt Versatz/Skalierung statt uebersprungen zu werden), der CFF -Umriss (Type-2-Charstrings als eigene Textform, TtxFont.CffGlyphs/Fonts/Type2CharstringText - alle Pfad-Operatoren inkl. der kompakten Formen hmoveto/hlineto/hhcurveto/vvcurveto/hvcurveto/vhcurveto werden gelesen [verifiziert an einer echten fonttools-Ausgabe], geschrieben wird ueber die vier einfachsten Operatoren + GlyphPath.ToCubic() [verlustfrei, anders als beim glyf-Schreiber keine Ablehnung]; Hint-/Unterprogramm-/Flex-Operatoren und endchar mit Akzentaufbau werden je Glyphe uebersprungen statt geraten) und die post-Namenstabelle (Format 2.0, extraNames nach derselben Regel wie fonttools' eigener Kodierer berechnet - mit der 258-Namen-Standard-Mac-Reihenfolge als Datentabelle in Fonts/TtxFont.cs). FED18 damit komplett, keine offenen Punkte mehr.		M
FED19	Metrikdateien AFM/PFM lesen und schreiben (gehoeren zu Type1). Fonts/AfmMetrics (Text, inkl. Kerning-Paare) + Fonts/PfmMetrics (binaerer GDI-Kopf + PostScript-Erweiterungskopf)		S
FED20	MCP + Pruefstand: convert_font (deckt sich mit FNT5), font_info, font_edit und ein Reiter fuer den Editor - (MCP-Werkzeuge font_info/font_edit in McpTools.All/dist/mcpb/manifest.json/McpToolGroups/Mcp.razor, dazu die eigenstaendige Pruefstand-Seite /schrifteditor als FED7)		S

v4-TXT · Klartext / Notepad

#	Notepad-Funktion	Stand bei dir	Aufw.
TXT1	Klartext-Editor als Pruefstand-Seite: oeffnen/bearbeiten/speichern, Zeilennummern, Suchen+Ersetzen, Gehe-zu-Zeile	M1320 (PlainText/TextLines + TextSearch in der Bibliothek, Seite /texteditor)	M
TXT2	Kodierung ERKENNEN — BOM + Heuristik mit Zuverlaessigkeitsangabe	M1313 (neues Blatt-Modul PlainText: SingleByteCharsets traegt die Zeichentabellen als Daten [Zero-NuGet - GetEncoding(1252)/CP850/MacRoman sind NICHT in-box], TextEncodingDetector erkennt ueber BOM oder Heuristik) + MCP detect_encoding. Ehrlich benannte Grenzen: ISO-8859-1 und Windows-1252 sind im Bereich 0xA0-0xFF Zeichen fuer Zeichen gleich und aus den Bytes NICHT unterscheidbar (die Erkennung meldet Windows-1252 mit hoechstens 0,75); kurzer Text traegt kaum Aussage; auf westeuropaeische Sprachen zugeschnitten	S
TXT3	Zeilenenden CRLF/LF/CR erkennen + umstellen (+ MCP-Werkzeug)	M1315 (PlainText/LineEndings + convert_line_endings) - gezaehlt wird je Zeilenumbruch, ein CRLF ist EIN Zeilenende	S
TXT4	Grosse Dateien (virtualisiertes Laden/Anzeigen ueber 100 MB, ohne alles im Speicher)	PlainText/LargeTextFile: Zeilen-Index ueber Byte-Versaetze, Zeilen einzeln nachladen, abschnittweise suchen mit Ueberlappung an den Grenzen. Grenze: ein Muster ueber mehr Zeilen als die Ueberlappung wird nicht gefunden	M
TXT5	Syntax-Hervorhebung + Klammerpaare	M1324 (JSON/XML/CSS/Markdown) + C# (gewoehnliche, woertliche, interpolierte und Roh-Zeichenketten, Doku-Kommentare)	M
Kodierung KONVERTIEREN — eine Datei in eine gewaehlte Zielkodierung schreiben (BOM ja/nein, Zeilenenden mitziehen), einzeln oder als Stapel ueber einen Ordner mit Bericht je Datei	M1315 (TextEncodingConverter/TextEncodingBatch + convert_encoding)	M	

v4-AG · Apogee-Rest (Druckvorstufe)

#	Vorstufen-Funktion	Stand bei dir	Aufw.
AG1	Transparenz-Flattener (Live-Transparenz verflachen fuer alte RIPs) — identisch mit AP5	M1321 (Editing/PdfTransparencyFlattener, MCP flatten_transparency)	L

#	Vorstufen-Funktion	Stand bei dir	Aufw.
AG2	Preflight-Fixup (korrigieren statt nur melden)	M1322 (Editing/PreflightFixup, MCP preflight_fix) - befundgetrieben: repariert nur, was der Preflight meldet	L
AG3	PDF/VT (variabler Datendruck) — die /DPart-Hierarchie ist seit M1167 da, es fehlen Metadaten + Regelwerk	PdfVtPartMetadata/PdfVtVersion beim Erzeugen, PdfVtCompliance beim Pruefen (setzt PDF/X-4 voraus). Grenze: ob die Blaetter inhaltlich je einen Datensatz treffen, bleibt ungeprueft	L
AG4	Normalizer (fremdes PDF in EINEM Lauf auf einen definierten Ausgabezustand bringen)	M1318 (Editing/PdfNormalizer, MCP normalize_pdf)	M

v4-ED · Editor — Paritaet zu den Vorbildern

#	Vorbild	Editor-Funktion	Stand bei dir	Aufw.
ED30	InDesign	Zeichenformat-Palette (eigene Absatz- UND Zeichenformate anlegen/verwalten/bearbeiten)	eigene Absatz- + Zeichenformate ueber Model/EditorStyleBook + Components/Editor/StylePalette.razor (ueber Kuonen.PDFTools.Styles.StyleCatalog, kein zweiter Formatmechanismus; getrennt von den 4 festen ED6-Formaten)	M
ED31	InDesign	Ausrichten & Verteilen + Duplizieren-und-versetzen — deckt LID5	M1347 (EditorAlignDistribute/EditorStepAndRepeat + Oberflaeche: 6 Ausrichtungen an Auswahl oder Seite, Verteilen nach Zwischenraum/Mitte, Step-and-Repeat)	M
ED32	InDesign	Objekt-Effekte-Palette (weiche Kante, Schein, Relief, Satin) — deckt LID1	Model/EditorEffectsPalette + Components/Editor/EffectsPalette.razor (ueber die LID1-Effekte, M1347)	M
ED33	InDesign	Story-Editor (Fliesstext-Fenster getrennt vom Layout, fuer langen Text)	Model/EditorStoryText + Components/Editor/StoryEditor.razor (Rahmenkette am Stueck bearbeiten, Kopf bekommt den Text, Folgerahmen werden vom Ueberlauf-Mechanismus wieder gefuellt)	M
ED34	InDesign	Verknuepfte statt eingebettete Bilder + Verknuepfungen aktualisieren/ersetzen/fehlende finden	Verknuepfungs-Palette (Model.EditorImageLinks + ImageFrame.ReplaceImageBytes)	L
ED35	InDesign	Live-Preflight-Panel im Editor (die Pruefung gibt es als Dienst, nicht als Palette)	Preflight-Palette (Model.EditorPreflightPanel, springt zur Seite)	M
ED36	InDesign	Softproof-/Ueberdrucken-Vorschau im Editor (ICC-Engine + Separationsvorschau sind da)	Softproof-Umschalter im Ansicht-Menue (Model.EditorSoftProof)	M
ED37	Acrobat	Kommentar-/Markup-Werkzeuge im Editor (Notiz, Hervorheben, Formen, Freitext) — das Annotations-Modul ist voll da	Playground/Services/EditorAcrobatSession + Components/Editor/AcrobatToolsPanel.razor, an einem PDF-Export des Editor-Dokuments (wie AP7)	M
ED38	Acrobat	Schwaerzen im Editor (RedactionEditor gibt es als Dienst, ED17)	selbe Sitzung/Feld - Markierungen sammeln, entfernen, anwenden	M

#	Vorbild	Editor-Funktion	Stand bei dir	Aufw.
ED39	Acrobat	Signieren im Editor (SignatureEditor gibt es als Dienst, ED18)	selbe Sitzung/Feld - Zertifikat laden, Flaeche aufziehen, sofort signieren	M
ED40	Acrobat	Zwei PDFs vergleichen im Editor (PdfComparer ist da)	selbe Sitzung/Feld - zweites PDF hochladen, je Seite zusammengefasste Abweichungen	S
ED41	Acrobat	Tag-Baum + Leseordnung bearbeiten (Barrierefreiheit) — die Editor-Seite von AP1	M1373	L
ED42	Acrobat	Fremdes PDF in-place editieren (ins gerenderte PDF klicken statt Import->Re-Render) = AP7	M1373	XL
ED43a	Word	Fliesstext-Modus: durchgehendes Textdokument, Seiten entstehen beim Tippen	M1492 - Palette „Fliesstext-Modus“ neben dem Rahmen-Modus: Absaetze, Auszeichnung, Tabellen, Bilder (M1381); in M1492 der Schreibstrich im Satzbild (FlowDocument.RecordTextPositions/FlowTextMap aus genau dem gezeichneten Umbruch; Klick, Pfeile, Pos1/Ende, Auswahl ueber Zeilen- und Seitenumbrueche; ein Klick auf die Stelle, die PageTextExtractor aus dem PDF liest, trifft genau dieses Zeichen), Fussnoten, Inhaltsverzeichnis und Formeln ([^a], [[toc]], $...$, $$...$$; neu FlowDocument.BuildWithTableOfContentsAndFootnotes) und die Uebernahme als echter Fluss-Container (LayoutDocument.AppendFlowContainers, in Rueckgaengig-Kette, PDF und .kdoc)	L
ED44	Word	Aenderungsverfolgung + Kommentare im Editor (TrackedText/EditorialNote sind da)	Playground/Model/EditorTrackChanges + Palette: Markup-Sicht (Einfuegung blau unterstrichen, Loeschung rot durchgestrichen, je mit Autor), Zaehlung, Kommentare anlegen/entfernen, Markup-PDF. Grenze des Modells: nur „alle annehmen"/„alle ablehnen" (kein selektives Annehmen); Notizen als Anhang am Dokumentende, nicht als Randnotiz	M
ED45	Word	Rechtschreibung beim Tippen (rote Wellenlinie) + Wort-/Zeichenzahl in der Statusleiste	Playground/Model/EditorProofing: der Text wird in Stuecke zerlegt (nichts geht verloren), unbekannte Woerter tragen die Wellenlinie; die Zaehlung steht im Fliesstext-Feld und in der Editor-Statuszeile. Grenze: die eingebaute Wortliste des SpellChecker ist kompakt - ohne Morphologie meldet sie auch gewoehnliche Woerter; eigene Woerter lassen sich mitgeben. Keine Grammatikpruefung, keine Korrekturvorschlaege	M
ED46	Word	Kopf-/Fusszeilen im Editor bearbeiten (der Stempel-Dienst HeaderFooterStamper ist da)	Playground/Model/EditorHeaderFooter + Palette: sechs Zonen, Groesse/Rand/erste Seitenzahl/Deckblaetter/Datumsformat, und eine Vorschau je Seite, die zeigt, was {page}/{pages}/{date} dort wirklich ergeben. Grenzen des Stempels: die Zeile liegt im Seitenrand (der Satzspiegel schrumpft nicht), je Zone eine Zeile ohne Auszeichnung, keine getrennten linken/rechten Seiten	M

#	Vorbild	Editor-Funktion	Stand bei dir	Aufw.
ED47	Excel	Tabellenblatt-Modus: Zellgitter, Formeleingabe (auch UEBER Blattgrenzen, Blatt2!A1), mehrere Blaetter	SpreadsheetPanel (Components/Editor/) - aus dem grossen Editor heraus erreichbar (Werkzeugleiste -> „Tabellenblatt...") UND weiter als eigenstaendige Seite /tabellenblatt; XlsxFormula.EvaluateAcrossSheets loest Blatt2!A1 auf (jedes Blatt wird zuerst fuer sich neu berechnet, ein unbekanntes Blatt wird als #FEHLER gemeldet, nie still 0). Ehrlich benannte v1-Grenze: nur Einzelzell-Bezuege - ein blattuebergreifender BEREICH (Blatt2!A1:B2) wird benannt abgelehnt	L
ED48	PowerPoint	Folien-Modus (Foliensortierung, Layouts, Platzhalter)	SlideSorterPanel (Components/Editor/) - aus dem grossen Editor heraus erreichbar (Werkzeugleiste -> „Folien...") UND weiter als eigenstaendige Seite /folien; SlideSorter.UpdateShapeGeometry/UpdateShapeText bearbeiten Stelle/Groesse/Text eines Rahmens (Platzhalters) direkt in der Oberflaeche. Ehrlich benannte v1-Grenze: ein Kind einer Gruppe ist ueber diesen Weg nicht einzeln erreichbar	L
ED49	PowerPoint	Praesentieren (Vollbild-Vorfuehrung, Referentenansicht, Notizen)	als eigenstaendige Pruefstand-Seite /praesentieren (PresenterView: Navigation + Notizen der RICHTIGEN Folie; Vollbild ueber die Fullscreen-API, aktuelle+naechste Folie als Referentenansicht)	M
ED50	Dreamweaver	HTML/CSS-Code-Editor mit Live-Vorschau (Split-View CodeAnsicht, /htmleditor) - jede Eingabe wandelt sofort ueber denselben HtmlConverter wie html_to_pdf		L
ED51	Dreamweaver	Projekt-/Site-Verwaltung (Dateibaum, lokale src/href-Verweise gegen den Projektordner pruefen, SiteFileTree/SiteLinkChecker)		M
ED52	Dreamweaver	Code-Vervollstaendigung + Validierung (HTML/CSS)	HTML: Model/HtmlTagCompletion + HtmlValidator (nicht geschlossene und falsch verschachtelte Elemente). CSS: Model/CssEditorSupport (Eigenschafts- und Wert-Vervollstaendigung, unbekannte Eigenschaft, nicht geschlossener Block) gegen die EINE Liste Html/CssPropertyCatalog im Kern - mit Drift-Waechter, der sie gegen den Quelltext des HtmlStyleResolver haelt, damit Vorschlag und tatsaechliches Koennen nicht auseinanderlaufen	M
ED53	Notepad	Klartext-Tab im Editor (siehe TXT1)	Components/Editor/PlainTextEditorPanel - der Klartext-Kern aus M1320 (Kodierung, Zeilenenden, Suchen/Ersetzen mit Zeile+Spalte, Gehe-zu-Zeile, Syntax, Klammerpaare) jetzt auch als Werkzeug im grossen Editor, kein zweiter Klartext-Weg gegenueber /texteditor	M

v4-FS · Datei- und Ordner-Werkzeuge (Nutzerwunsch)

#	Punkt	Stand	Aufw.
FS1	file_info — alles, was der Eigenschaften-Dialog zeigt: Groesse, alle drei Zeitstempel (erstellt / geaendert / zugegriffen), die Attribute (schreibgeschuetzt, versteckt, System, Archiv, komprimiert, verschluesselt), ob es eine Verknuepfung / Abzweigung ist und wohin sie zeigt, und die vollstaendige Versionsressource: Dateibeschreibung, Datei- und Produktversion, Produktname, Firma, Copyright, Originaldateiname und Sprache. Gemessen an einer echten Systemdatei: alle acht Felder kommen ueber FileVersionInfo eingebaut zurueck, wortgleich zum Windows-Dialog	M1424: file_info liefert alle acht Felder der Versionsressource, die drei Zeitstempel, die sechs Attribute und das Ziel einer Verknuepfung. Nicht dabei, je mit Grund: die Groesse auf dem Datentraeger (GetCompressedFileSize), die Dateiart im Klartext samt "Oeffnen mit" (steht in der Registrierung, nicht in der Datei) und die Vorgaengerversionen (Schattenkopien)	M
FS2	Attribute und Zeitstempel setzen — schreibgeschuetzt / versteckt / System / Archiv setzen und loeschen, Zeitstempel aendern; einzeln oder rekursiv	M1424: set_attributes und set_timestamps, einzeln oder rekursiv, ueber den geteilten Lauf (FS7). Steht der Wert schon so da, meldet der Bericht schon richtig statt geaendert - bei tausend Dateien ist genau das die Zahl, die zaehlt. Versteckt/System/Archiv gibt es unter Unix nicht: dort benannt, nicht nachgebildet (ein fuehrender Punkt waere eine Umbenennung, also ein ganz anderer Eingriff)	S
FS3	Berechtigungen LESEN — unter Windows die ACL (je Eintrag Konto, Rechte, Erlauben/Verweigern, geerbt ja/nein), unter Linux und macOS der POSIX-Modus samt Besitzer und Gruppe. Nie geraten: auf dem jeweils anderen System wird die fehlende Form benannt, nicht nachgebildet	M1424: file_permissions - Windows Besitzer, Gruppe und jeder Zugriffseintrag (Konto, Rechte, erlaubt/verweigert, geerbt); Unix der POSIX-Modus. Die auf dem System fehlende Form wird gesagt, nie nachgebildet. Nicht dabei: die Pruefungsliste (SACL/Auditierung, Privileg SeSecurity), und unter Unix Besitzer und Gruppe - die stat-Struktur liegt je System und Architektur anders, sie zu raten waere schlimmer als sie wegzulassen	M
FS4	Berechtigungen SETZEN — Rechte hinzufuegen, entfernen, ersetzen; Vererbung ein- und ausschalten; einzeln oder rekursiv. Mit Trockenlauf als Vorgabe: ein rekursiver Rechte-Eingriff ist der gefaehrlichste Aufruf im ganzen Werkzeugkasten — er zeigt erst, was er tun wuerde, und tut es erst auf ausdrueckliche Anweisung	M1424: set_permissions - hinzufuegen, entfernen, ersetzen, Vererbung ein-/ausschalten, einzeln oder rekursiv. Trockenlauf ist die Vorgabe (erst apply=true fuehrt aus), und es braucht zwei Freigaben zugleich: KUONEN_PDFTOOLS_ENABLE_FS_WRITE=1 UND erhoehte Rechte - das ist strenger als Windows selbst (dort darf ein Nutzer die Rechte an einer eigenen Datei aendern) und ausdruecklicher Nutzerwunsch	L
FS5	Besitzer aendern — einzeln oder rekursiv. Ehrlich vorweg: unter Windows braucht das das Privileg SeTakeOwnership (also erhoehte Rechte), unter Unix in aller Regel root; und einen verwalteten chown gibt es in .NET nicht — das laeuft ueber DllImport("libc"), wie es SpoolerStart.AktuelleKennung (M1414) fuer geteuid schon tut. Fehlt das Recht, muss das Werkzeug es sagen, statt still nichts zu tun	M1424: set_owner, gleiche Schranke und gleicher Trockenlauf. Windows nennt bei fehlendem Recht das Privileg SeTakeOwnership beim Namen. Unter Unix laeuft es ueber DllImport("libc") und nimmt Nummern (uid bzw. uid:gid): einen Namen aufzuloesen braeuchte dieselbe nicht verwaltete passwd-Struktur, deshalb wird ein Name abgelehnt statt geraten	M

#	Punkt	Stand	Aufw.
FS6	Symbole aus beliebigen Dateien	M1424/M1425: file_icon deckt die drei Wege (Symboldatei .ico/.cur/.icns, eingebettetes Symbol einer Ressourcendatei, Zuordnungssymbol einer Dateiart aus der Registrierung), entschieden an der Signatur, nicht an der Endung. M1425 hat den Rest geschlossen: der Store-Paket-Verweis (@{Paket?ms-resource://...}, gemessen bei .txt/.png/.jpg/.mp3) wird ueber SHLoadIndirectString von der Shell aufgeloest - zurueck kommt ein echtes, lesbares PNG unterhalb von WindowsApps; die negative Zahl ist eine Ressourcen-ID und wird ueber die neu herausgegebenen Gruppen-IDs gesucht (imageres.dll,-68 = das Symbol von .bat), nicht als Position geraten; und den ausgelagerten .mun-Ressourcen wird gefolgt (System32\imageres.dll ist ein 12-KB-Rumpf, die 24 MB liegen daneben - ohne diesen Schritt liefe der ganze Weg fuer Systemdateien ins Leere). Grenzen, die Windows setzt: der Paket-Verweis liefert eine Fassung (die der laufenden Sitzung), nicht alle im Paket abgelegten Groessen; ausserhalb von Windows gibt es weder Zuordnung noch Paket-Verweis und beides wird benannt abgelehnt	M
FS7	Rekursion, Filter und Bericht als durchgaengiges Prinzip — jede aendernde Aktion nimmt dieselben Angaben (Muster, Tiefe, nur Dateien / nur Ordner) und liefert je Eintrag, was geschah: geaendert, schon richtig, uebersprungen, fehlgeschlagen mit Grund. Ein stiller Sammel-Lauf ueber tausend Dateien waere wertlos	M1424: DateiLauf - Muster, Tiefe und Art fuer jede aendernde Aktion, dazu der Bericht je Eintrag (geaendert / schon richtig / uebersprungen / fehlgeschlagen mit Grund). Die Vorgabe ist der sichere Fall: Tiefe 0 = nur der genannte Pfad; das Muster filtert die Kinder, nie den ausdruecklich genannten Pfad. Ein Wurf im Einzelfall wird zum Eintrag, nicht zum Abbruch	M
FS8	Signatur einer Programmdatei lesen — der Reiter „Digitale Signaturen“: wer signiert hat, mit welchem Verfahren, wann gegengezeichnet. Gemessen: den Unterzeichner liefert .NET eingebaut (CN=Microsoft Corporation kam an Ihrer Datei zurueck). Ehrlich die Grenze: das ist das blosse Zertifikat — die Pruefung (Kette bis zur Wurzel, Zeitstempel, Sperrliste, stimmt der Dateiinhalt noch zur Signatur) ist etwas anderes und braucht WinVerifyTrust ueber P/Invoke. Ein Werkzeug, das „signiert von X“ meldet, ohne geprueft zu haben, waere gefaehrlicher als keines	M1424: file_signature liest Antragsteller, Aussteller, Gueltigkeit, Verfahren und Fingerabdruck. Die Grenze ist der Punkt: nichts davon ist geprueft - weder Vertrauenskette noch Zeitstempel noch Sperrliste, und nicht, ob der Dateiinhalt noch zur Signatur passt; das braeuchte WinVerifyTrust. Der Bericht sagt es in jeder Ausgabe. Gemessen: die Dateien in System32 sind katalogsigniert, tragen also gar keine eingebettete Signatur - daraus "nicht signiert" zu machen waere eine falsche Aussage ueber eine signierte Datei	M

INF · Betrieb und Auslieferung (Nutzerwunsch)

#	Punkt	Stand
INF5	Ein Dienst darf fehlen duerfen: der Pruefstand zeigt einen Reiter nur, wenn die zugehoerige Instanz wirklich antwortet	fuer den Spooler (M1400: Lebendmeldung /gesundheit, kurzes Zeitlimit, Tab verschwindet sonst) · fuer den Noten-Dienst

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

#	Funktion	Stand bei dir	Aufw.
MUS1	Notenmodell: Partitur -> Stimmen -> Takte -> Ebenen -> Noten/Pausen; Tonhoehe, Dauer, Vorzeichen, Schluessel, Tonart, Taktart, Haltebogen, Balken, N-tolen, Vorschlagsnoten	Music (Score/Part/Measure/Voice/Note/MusicRest, dazu Pitch/NoteDuration/Clef/KeySignature/TimeSignature/TupletSpan). Ein reines Datenmodell - es rechnet und zeichnet nichts, wie QrBill und Invoicing. Stammton und Versetzung bleiben getrennt, sonst waeren cis und des nicht mehr unterscheidbar	M
MUS3	Stech-Regeln (der Kern): Notenkopf-Positionen, Halsrichtung und -laenge, Balkengruppen und ihre Neigung, Hilfslinien, Vorzeichen-Stapelung, Kollisionsvermeidung und die horizontale Verteilung	Music/ScoreEngraver + EngravingRules. Kopfposition aus Schluessel und Stufe (vier Schluessel), Halslaenge eine Oktave, Balken bis Zweiunddreissigstel mit begrenzter Neigung und Teilbalken, Hilfslinien, Vorzeichen-Stapelung, Ausweichen der Sekunde im Akkord, dauerabhaengiger Abstand, Punktierungen, Fahnen, verkleinerte Vorschlagsnoten. Seit M1404: mehrere Stimmen je Takt auf derselben Zeile mit Halsrichtung nach der Stimmenrolle (nicht nach der Lage - daran unterscheidet sich echter Mehrstimmensatz von „zweimal einstimmig uebereinander"), Haltebogen ueber Taktstrich und Zeilenumbruch, Instrument-Vorgabeschluessel und Tonumfangspruefung je Takt. Offen, je gemeldet: Bindeboegen, N-tolen-Klammer, Halsverlaengerung zur Mittellinie, Balkenlage auf Linienmitte	XL
MUS9	MusicXML lesen (.musicxml und gepackt .mxl) — der Weg herein aus jedem anderen Programm	MusicXml/MusicXmlReader (M1404), neues Blatt-Modul mit der einen Kante auf Music. Beide Erscheinungsformen (die ZIP-Signatur entscheidet, nicht die Endung), part-list, Schluessel/Tonart/Taktart, Tonhoehe mit Stammton und Versetzung, Dauer mit Punktierung, Pausen, Akkorde, Haltebogen, Balken, Vorzeichen, Vorschlagsnoten, N-tolen. Die zwei Fallen des Formats sind je mit Gegenbeleg getestet: duration steht in divisions (je Datei frei gewaehlt, nicht in Vierteln), und backup setzt die Zeit im Takt zurueck. XXE-fest. Gegen music21 verifiziert. Offen: score-timewise wird benannt abgelehnt; Liedtext/Dynamik/Boegen/Wiederholungen werden gemeldet, nie still verschluckt	L
MUS10	MusicXML schreiben — der Weg hinaus; damit ist der Rundlauf zu, wie bei DOCX/XLSX	MusicXml/MusicXmlWriter (M1404) - unkomprimiertes score-partwise 4.0, Teiler 3360 (so gewaehlt, dass jede Dauer des Modells ganzzahlig wird), backup zwischen mehreren Stimmen. music21 liest die eigene Ausgabe deckungsgleich zum Original zurueck, inklusive zweier gleichzeitiger Stimmen im selben Takt. Offen: die gepackte .mxl wird gelesen, aber nicht geschrieben	M
MUS11	MIDI lesen und schreiben (Standard MIDI File 0/1) samt Anbindung an das Notenmodell	Der Codec (Midi, M1372) und die Bruecke zur Notation (MidiNotation, M1404): Note-An/Note-Aus paaren, quantisieren (Massstab ist die Kopf-Aufloesung, nicht eine feste Tickzahl - daran faellt ein naiver Wandler auf), Notenwerte mit Punktierung, Takte aus dem Taktart-Meta-Ereignis, Enharmonik aus der Tonart (MIDI 61 ist cis oder des - geraten wird nie), Stimmen je Spur/Kanal, Schluessel aus dem Tonumfang. Gegen mido verifiziert. Offen, je gemeldet: kein Haltebogen ueber den Taktstrich (der Ton beginnt den naechsten Takt ganz und klingt damit spaeter als gespielt), nur eine Stimme je Takt, keine N-tolen, kein Tempo und keine Dynamik; SMPTE-Zeitcode wird benannt abgelehnt	M

#	Funktion	Stand bei dir	Aufw.
MUS15	MCP-Werkzeuge und ein Pruefstand-Reiter	(M1404) midi_to_pdf, musicxml_to_pdf, midi_to_musicxml und musicxml_info - Letzteres sagt vor dem Wandeln, was verlorengehen wird. midi_to_pdf und musicxml_to_pdf geben beide Berichte aus (Uebersetzung und Notenstich); sonst bliebe unsichtbar, dass ein Ton wegen des fehlenden Haltebogens spaeter klingt als gespielt. Damit ist die Fehlerklasse geschlossen, die hier dreimal auftrat (IMG1/IMG22/IMG27): gebaut, aber nicht ansteuerbar	S

MUS13 · Der Noten-Editor im Einzelnen — 5 von 14

#	Scheibe	Stand
MUS13.1	Eigene Seite /noteneditor mit eigenem Knopf neben dem Editor	M1403
MUS13.2	Mehrere Spuren mit Instrument je Spur (39 Instrumente), Systemgruppen	M1404
MUS13.3	Rechenkern MusicEditing (diatonisch verschieben, Zeitwert, Takteinteilung) — rein und testbar, wie TableGridEditing	M1403
MUS13.4	Vorschau auf den bedruckten Bereich zugeschnitten (eine A4-Seite ist zu ueber 90 % weiss)	M1403
MUS13.5	Import: MusicXML, MIDI, PDF und jedes Bildformat	M1404

v4-PRN · Virtueller PDF-Drucker (Nutzerwunsch)

#	Drucker-Funktion	Stand bei dir	Aufw.
PRN1	IPP-Server (HTTP + Get-Printer-Attributes/Validate-Job/Print-Job/Get-Jobs/Cancel-Job): der Pruefstand meldet sich als Drucker „Kuonen PDF“ und nimmt den Job entgegen	Printing/IppPrintService (netzfreier Kern) + Playground/IppPrintServer (POST /ipp/print). Seit M1391 zusaetzlich Create-Job/Send-Document (RFC 8011 §4.2.2/§4.2.3) - die zweiphasige Einreichung, bei der Cancel-Job zwischen den zwei Anfragen wirklich greift (der Auftrag steht dazwischen als Pending, noch ohne Dokument/Verarbeitung); Print-Job bleibt einphasig-synchron und damit unveraendert praktisch nie abbrechbar (dafuer nutzt ein Client den zweiphasigen Weg). Ausserdem opt-in ipps:// (IppTlsCertificate + KUONEN_PDFTOOLS_IPPS_PORT/KUONEN_PDFTOOLS_TLS_CERT_PATH) und die bereits vorhandene globale Basic-Auth-Schranke (AUTH1) deckt /ipp/print wie jeden anderen Pfad. Bekanntmachung im Netz siehe PRN3. Seit M1392 zusaetzlich Get-Job-Attributes (RFC 8011 §4.3.4 - die Auskunft ueber EINEN Auftrag) sowie die volle Auftragssteuerung Hold-Job/Release-Job/Restart-Job/Purge-Jobs (RFC 8011 §4.3.6-§4.3.9, neu IppJobState.Held + IppPrintQueue.Hold/Release/Restart/Purge, bewusst enge Uebergangs-Waechter: Hold nur aus Pending, Restart nur aus Completed) - job-state-reasons traegt fuer einen zurueckgestellten Auftrag jetzt job-hold-until-specified (aus dem vorhandenen Zustandsmodell abgeleitet, nicht erfunden). Pause-Printer/Resume-Printer bleiben ehrlich unimplementiert (der Dienst arbeitet synchron, kein Zustand „angehalten“)	L
PRN2	Spooler + Oberfläche: Warteschlange mit Absender/Zeit/Seitenzahl/Status, Vorschau, Herunterladen, Loeschen, Aufbewahrungsdauer	Printing/PrintSpoolRetention (reine Ablaufregel, ohne Uhr) + Playground/Services/PrintSpoolService (baut auf dem Ausgabeordner auf, den IppPrintServer bereits fuellt - IppPrintQueue hat weder Persistenz noch eine Loeschfunktion) + Seite /druckerwarteschlange. Seit M1391 zeigt die Warteschlange zusaetzlich die ECHTE Herkunft (Absender-Adresse + optional requesting-user-name, aus einer Sidecar-Datei neben der Auftrags-PDF, Playground/Model/PrintSpoolOrigin) - Absender bleibt der frei waehlbare job-name, Herkunft ist die verlaessliche Angabe	M
PRN4	Rueckgabe an das ausloesende Geraet: Link/QR-Code direkt nach dem Druck, wahlweise Ablage in einem ueberwachten Ordner oder Netzlaufwerk	Playground/Model/PrintSpoolReturnLink (Link + QR-Code ueber die vorhandenen Barcode-Bausteine, ueber den Barcode-Leser rueckgelesen geprueft) + PrintSpoolService.TryDropToWatchedFolder (Kopie in einen konfigurierbaren Ordner). Grenze: ein Netzlaufwerk muss bereits als lokaler Pfad erreichbar sein (kein UNC-Mount mit eigenen Zugangsdaten)	M
PRN5	PostScript-Port-Variante fuer Alt-Programme, die zwingend einen lokalen Drucker wollen: generischer Windows-PostScript-Treiber + eigene PPD -> Datei-Port -> der vorhandene PostScriptToPdf (M954). Kein eigener Treibercode, keine Signierung	Printing/PrinterPpd (PPD aus IppPrinterAttributes abgeleitet - keine Widerspruche zwischen PPD und IPP-Selbstbeschreibung) + Playground/Services/PostScriptDropFolder (opt-in ueber KUONEN_PDFTOOLS_PS_DROPFOLDER, wandelt ueber den vorhandenen PostScriptToPdf). Einrichtung in docs/drucker-einrichten.md. Grenze: gegen einen echten Windows-Treiber auf dieser Maschine nicht pruefbar (kein Windows-Drucksystem im Testlauf) - geprueft ist die PPD-Syntax gegen die Adobe-Spezifikation	M

#	Drucker-Funktion	Stand bei dir	Aufw.
PRN6	Nicht-PDF-Jobs wandeln: PWG-Raster/URF (iOS/AirPrint) und PCL in PDF ueberfuehren	beide Haelften vollstaendig; PWG seit M1392 gegen die amtliche CUPS-Referenz fremdverifiziert (Fixture aus cupsfilter+rastertopwg im Container, im Testlauf bleibt kein Fremdwerkzeug zurueck - sie deckte drei Fehler auf: die Auswertung von cupsCompression, den fehlenden Farbraum K und die vertauschten Zweige der Lauflaengen-Regel). Fuer PCL/HP-GL/2 gibt es weiterhin kein Orakel (kein GhostPCL). (a) PWG-Raster + Apple URF (Printing/PwgRaster + PwgRasterToPdf) - Grau/sGray/Schwarz-Weiss (1/8/16 Bit) und RGB (sRGB/AdobeRGB, 8/16 Bit); zusaetzlich CMYK (ohne Farbprofil ueber die schlichte multiplikative Rechnung R=(255-C)·(255-K)/255 nach RGB gewandelt - ein PWG-Rasterstrom traegt selbst kein Farbprofil, und das eingebaute generische CMYK-Profil waere nicht genauer) und CIELab (ueber die genormte Lab->XYZ->sRGB-Rechnung, Referenzweiss D65, ebenfalls ohne Geraeteprofil). Ein 16-Bit-Kanal wird auf das 8-Bit-Ausgabemodell gebracht, indem das hoechstwertige Byte genommen wird (nicht das niederwertige - sonst fiele z. B. 0x0100 faelschlich auf 0). Sonderfarben und alles darueber hinaus bleiben benannt abgelehnt, nie geraten; Apple URF hat keine amtliche Nummerierung fuer diese Farbraeume - die Erweiterung setzt die schon lokal gewaehlte Nummerierung (Grau=1/RGB=2) um CMYK=3/CIELab=4 fort. (b) PCL5 (Printing/PclToPdf + PclInterpreter) - Seitensteuerung (Groesse/Ausrichtung/Zeilenabstand/Formfeed), Cursor-Positionierung (Dezipunkte + Bildpunkte), Text auf den Standard-14-Schriften (Groesse/Stil/Strichstaerke/Familie/Pitch), Raender (linker/rechter/oberer Rand, Textlaenge), Symbolsaetze fuer 0x80-0xFF, HP-GL/2 als eingebetteter Vektor-Modus und Rastergrafik (unkomprimiert, PCL-Lauflaengenkodierung, TIFF-PackBits, Delta-Row) - ueber die vorhandene PageCanvas-Zeichen-API, kein zweiter Zeichenweg (dasselbe Vorbild wie PostScriptToPdf). Seit der letzten Scheibe zusaetzlich: eingebettete Bitmap-Ladeschriften (ESC*c#D/ESC)s#W/ESC*c#E/ESC(s#W/ESC(<ID>X, siehe Printing/PclSoftFont) - jede geladene Glyphe wird als Bild-XObject gezeichnet, nicht als eingebautes Textzeichen - und ein echter Randumbruch: erreicht der Text den rechten Rand (ESC&a#M), geht es in der naechsten Zeile weiter, sofern der Zeilenende-Modus (ESC&s#C) das nicht abschaltet. PCL-XL (PCL 6, eine voellig andere binaere Sprache) wird an seiner Kennung erkannt und benannt abgelehnt, statt unvollstaendig gelesen zu werden. Ehrlich benannte v1-Grenzen: eine Umriss-Ladeschrift (Intellifont/TrueType, Kopf-Format 10/11/15/16) wird abgelehnt (nur Bitmap-Kopfformate 0/20 werden angenommen); bei den Zeichendaten nur das unkomprimierte Deskriptor-Format 4/Klasse 1; eine Bitmap-Ladeschrift traegt im gelesenen Deskriptor-Teil keine eigene Aufloesungsangabe - die Glyphen werden mit einer festen 300-dpi-Naeherung platziert; der Randumbruch bricht am erreichten Zeichen, nicht an einer Wortgrenze; Delta-Row-Kompression braucht eine vorher gesetzte Rasterbreite. Kein unabhaengiges Orakel fuer PWG/URF/PCL verfuegbar (kein CUPS/GhostPCL auf dieser Maschine) - Fixtures von Hand nach Spezifikation gebaut. HP-GL/2 vervollstaendigt: alle drei Teile - PWG/URF, PCL5 und der eingebettete HP-GL/2-Vektormodus. (a) PWG-Raster + Apple URF (Printing/PwgRaster + PwgRasterToPdf) - Grau/sGray/Schwarz-Weiss (1/8/16 Bit)	L

#	Drucker-Funktion	Stand bei dir	Aufw.
		und RGB (sRGB/AdobeRGB, 8/16 Bit); zusaetzlich CMYK (ohne Farbprofil ueber die schlichte multiplikative Rechnung R=(255-C)·(255-K)/255 nach RGB gewandelt - nicht weil Printing die ICC-Engine architektonisch nicht importieren duerfte, sondern weil ein PWG-Rasterstrom kein Farbprofil traegt und das generische CMYK-Profil selbst nur ein analytisches Modell ist: farbmanagement-echt wird die Wandlung erst mit dem Profil der Druckbedingung, das der Aufrufer beisteuern muesste) und CIELab (ueber die genormte Lab->XYZ->sRGB-Rechnung, Referenzweiss D65, ebenfalls ohne Geraeteprofil). Ein 16-Bit-Kanal wird auf das 8-Bit-Ausgabemodell gebracht, indem das hoechstwertige Byte genommen wird (nicht das niederwertige - sonst fiele z. B. 0x0100 faelschlich auf 0). Die uebrigen CUPS-Farbraeume (CMY/YMC/YMCK/...) bleiben benannt abgelehnt, nie geraten. (b) PCL5 (Printing/PclToPdf + PclInterpreter) - Seitensteuerung (Groesse/Ausrichtung/Zeilenabstand/Formfeed), Cursor-Positionierung (Dezipunkte + Bildpunkte), Text auf den Standard-14-Schriften (Groesse/Stil/Strichstaerke/Familie/Pitch) mit Symbolsaetzen fuer 0x80-0xFF (Roman-8/PC-8/PC-850/Windows-1252/ISO-8859-1), Raendern, und Rastergrafik (unkomprimiert, PCL-Lauflaengenkodierung, TIFF-PackBits, Delta-Row-Kompression Art 3) - ueber die vorhandene PageCanvas-Zeichen-API, kein zweiter Zeichenweg. PCL-XL (PCL 6, eine voellig andere binaere Sprache) wird an seiner Kennung erkannt und benannt abgelehnt, statt unvollstaendig gelesen zu werden; eingebettete Ladeschriften bleiben uebersprungen und gemeldet. (c) HP-GL/2 (Printing/HpglInterpreter, seit M1391 ueber die urspruengliche 7-Farben-/10x7,5-Zoll-Naeherung hinaus ausgebaut): echte Stiftfarben ueber PC/NP statt der festen Palette; die Bezugspunkte P1/P2 ueber IP/IR gesetzt und SC bezieht sich darauf statt auf eine angenommene Flaeche (behebt die Verzerrung bei abweichender Zeichenflaeche), dazu ein Eingabefenster IW (was ausserhalb liegt, wird nicht gezeichnet); Beschriftungsgroesse absolut/relativ (SI/SR), Schreibrichtung (DI/DR) und Ankerpunkt (LO, 1..9) - wirken auf Groesse und Platzierung der Beschriftung, nicht auf eine echte Drehung der Glyphen selbst (die braeuchte eine neue PclToPdf-Faehigkeit); Boegen (AA/AR, als Sehnenkette angenaehert), Kreissegmente (WG gefuellt/EW umrandet) und eine gestrichelte Linienart (LT, ueber sechs fest hinterlegte Strich/Luecke-Muster, das Muster beginnt bei jeder Strecke neu). Ehrlich benannt bleiben unerreichbar, solange nur der Interpreter (nicht der Renderer PclToPdf) angefasst werden darf: LA (Linienenden/-verbindungen) braeuchte Linienkappen-Unterstuetzung im Renderer, FT (Schraffur-Fuellarten) braeuchte Muster-Fuellungen - beide werden weiterhin gemeldet, nie geraten; ein aufgezeichnetes Vieleck (PM/FP/EP) wird von IW nicht beschnitten; SC nur Typ 0 (linear). Kein unabhaengiges Orakel fuer PWG/PCL/HP-GL/2 verfuegbar (kein CUPS, kein GhostPCL auf dieser Maschine) - Fixtures von Hand nach Spezifikation gebaut.	

#	Drucker-Funktion	Stand bei dir	Aufw.
PRN7	Zugang: Authentifizierung, IP-Beschraenkung, Groessen-/Mengenkontingent je Absender	Printing/IppAccessPolicy (IP-CIDR-Bereiche, Einzelgroessen-Deckel, Byte-/Auftrags-Kontingent im gleitenden Fenster) und verdrahtet: IppPrintServer.Handle(bytes, remoteIp) fragt die Regel vor jedem Druckauftrag (Print-Job UND Send-Document), ein abgelehnter Auftrag wird gar nicht erst angelegt; Program.cs baut die Regel opt-in aus vier Umgebungsvariablen (KUONEN_PDFTOOLS_IPP_ALLOWED_CIDRS/_MAX_JOB_BYTES/_QUOTA_BYTES/_QUOTA_JOBS/_QUOTA_WINDOW_HOURS) - ohne sie unveraendert unbeschraenkt. Der Grund kommt als auswertbarer DenyReason zurueck (nicht als Textvergleich) und waehlt den IPP-Status. Seit M1391: der Verbrauch je Absender wird als JSON neben den Auftraegen gesichert und beim Start wieder geladen - ueberlebt einen Neustart (ein anderer Ausgabeordner faengt bei null an); die Anmeldung laeuft ueber das vorhandene BasicAuthGate (AUTH1), das VOR jeder Anfrage inkl. /ipp/print sitzt - kein zweites, IPP-eigenes Verfahren. Ehrlich benannte Grenze: Absender bleibt die Adresse - IPP traegt keine verlaessliche Absenderkennung (requesting-user-name setzt der Client frei und dient nur der Anzeige, siehe PRN2)	S
PRN8	Druckoptionen uebernehmen aus den IPP-Job-Attributen	M1327 (Printing/IppJobOptions) - Kopien und Ausrichtung werden angewandt, Duplex/Farbe/Format nur mitgefuehrt	M

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

#	Scheibe	Stand
PRN3.1	mDNS-Datensaetze bauen (_ipp._tcp, AirPrint-TXT-Schluessel) — netzfreier Kern	M1390 - von dnspython gegengelesen
PRN3.2	UDP-Multicast-Versand im Spooler	M1390
PRN3.3	Die Bekanntmachung muss den Container verlassen	M1409 - gemessen, nicht beschrieben (Lauscher auf dem Wirt, mit unserem eigenen Mdns-Leser). Positivkontrolle ohne Docker: 3 Pakete. Bruecken-Netz: nichts (kuendigt 172.17.0.2 an). network_mode: host: nichts - unter Docker Desktop ist das „Wirtsnetz“ die VM (192.168.65.6), nicht das LAN. macvlan: nichts, die Adresse war vom Wirt nicht erreichbar. Wirtsseitiger Ankuendiger: 4 Pakete, Adresse erreichbar. WSD (239.255.255.250:3702) verhaelt sich gleich: aus dem Container nichts, vom Wirt 5 von 5. Folge: Linux-Server -> Wirtsnetz oder macvlan (erwartet, mangels Maschine nicht messbar); Windows-/macOS-Wirt -> wirtsseitiger Ankuendiger (scripts/netzweg-ankuendiger.cs, baut die Pakete aus Mdns/IppServiceAnnouncement - kein zweiter Paketbau). Ein mDNS-Reflektor wurde bewusst NICHT gebaut: er spiegelt das Paket, aber die Adresse darin bleibt 172.17.0.2 - das Geraet faende den Drucker und kaeme nicht an ihn heran

#	Scheibe	Stand
PRN3.4	IPP Everywhere vollstaendig (PWG 5100.14): printer-device-id, ipp-features-supported, media-col-database, printer-icons, printer-strings-uri	M1409 - das ist der „Treiber“: bei IPP Everywhere gibt es keinen, das System baut die Warteschlange aus der Selbstauskunft. Neu die IPP-Sammlungen nach RFC 8010 §3.1.6 (schreiben und lesen - ohne sie liesse sich media-col-database gar nicht ausdruecken), die Pflichtangaben samt Deckungsnachweis, sowie Geraetesymbol und Sprachdatei gerechnet statt mitgeliefert (kein Binaerbestand im Repo). Ehrlich offen: ipptool vermisst weiterhin printer-make-and-model, printer-info, printer-location, printer-more-info, printer-up-time und media-col-default
PRN3.5	Probing, Konfliktbehandlung und Abmelde-Pakete (RFC 6762 §8.1/§10.1)	M1409 - MdnsProbing (dreimal sondieren im Abstand von 250 ms, Vorschlag im Authority-Abschnitt), MdnsRecordOrder (der lexikografische Vergleich aus §8.2.1 - ohne Cache-Flush-Bit, ohne Lebensdauer, Namen kleingeschrieben), MdnsGoodbye (§10.1) und MdnsResponder (Bekannte-Antwort-Unterdrueckung §7.1, QU-Bit §5.4). Der Fund beim Bauen: die erste Fassung wertete JEDEN Datensatz als Konflikt - ein PTR auf _ipp._tcp.local ist aber ein geteilter Eintrag, ein zweiter Drucker dort ist der Normalfall; der Drucker haette sich im erstbesten Netz mit einem zweiten endlos umbenannt. Jetzt zaehlen nur die eindeutigen Datensaetze. Offen: die Drosselung nach §9 (>15 Konflikte in 10 s), der Konflikt im laufenden Betrieb, die Zufallsverzoegerung 20-120 ms fuer geteilte Datensaetze (§6) und die Fortsetzung einer Known-Answer-Liste ueber Pakete (§7.2)
PRN3.6	Windows: WSD (Web Services on Devices) als zweiter Weg	M1409 - neues Blatt-Modul Wsd (importiert nichts, wie Mdns): alle sechs Nachrichten (Hello/Probe/ProbeMatches/Resolve/ResolveMatches/Bye) schreiben und lesen, Gerätebeschreibung, WS-MetadataExchange, Antwortregel mit APP_MAX_DELAY 500 ms (Zeit und Zufall werden hineingereicht). Fassung 2005/04, nicht die spaetere OASIS-Fassung - Windows spricht bis heute die alte. Die Geraetekennung wird aus dem Rechnernamen abgeleitet, nicht gewuerfelt: sonst legte Windows bei jedem Neustart einen neuen Drucker an. Fremdverifiziert gegen die Python-Bibliothek WSDiscovery in BEIDEN Richtungen (ihre Nachrichten als Fixture-Konstanten im Repo, kein Fremdwerkzeug im Testlauf), quergelesen an wsdd. Der unterscheidende Fall: die fremde Probe nennt den Druckertyp unter dem Praefix zzz statt wprt - wer Typen samt Praefix als Zeichenkette vergleicht, antwortet einem echten Client nie. Ehrlich benannte Grenze: WSD ist hier nur der Erkennungsweg. WS-Print (die Druckseite mit CreatePrintJob/SendDocument) ist NICHT gebaut - gedruckt wird ueber IPP, und die Metadaten verweisen dorthin. Ebenfalls offen: IPv6, WS-Security, WS-Eventing

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

#	Lightroom-Funktion	Stand bei dir	Aufw.
LR2	Grundeinstellungen: Belichtung, Kontrast, Lichter, Tiefen, Weiss, Schwarz, Klarheit, Dunst entfernen, Dynamik, Saettigung	Photo/BasicAdjustments (M1404). Belichtung als Multiplikation in linearem Licht (auf den sRGB-Zahlen gerechnet saehe es bei kleinen Werten fast richtig aus - genau das war einer der gefundenen Fehler); die vier Tonwertbereiche ueber eine Maske aus der wahrgenommenen Luminanz (mittleres Grau liegt wahrgenommen bei 0,50, linear bei 0,216 - mit der linearen Maske zog „Tiefen" bis weit in die Mitteltoene); Weissabgleich in Mired statt linear in Kelvin (linear wird der Rot-Faktor jenseits ~18 300 K negativ). Offen: Kontrast dreht um einen festen Drehpunkt (kein S-Kurven-Modell), Klarheit ist Unscharf-Maskieren mit Box-Filter, Dunst ohne Guided-Filter-Verfeinerung	L
LR3	Tonwertkurve (Punkt + parametrisch je Kanal), HSL/Farbmischer, Farbkalibrierung	Photo/ToneCurve + Photo/ColorMixer (M1404). Die Kurve ist monoton (die uebernommene Fassung schwang ueber und erzeugte bei gewoehnlichen Stuetzpunkten an 16 Stellen eine echte Tonwertumkehr), alle vier parametrischen Regler wirken ueber ihre ganze Bahn, und die acht Farbbereiche sitzen an den echten Farbtonstellen (gleichmaessig alle 45 Grad verteilt bekam ein reines Blau von seinem Regler nur zwei Drittel). Neutrale Bildpunkte fallen nicht mehr ins Rot-Band. Offen: die Farbkalibrierung; die Zugehoerigkeit haengt allein am Farbton, nicht zusaetzlich an Helligkeit/Saettigung	L
LR6	Transformieren: Perspektive/Ausrichten (Upright), Zuschnitt, Drehung, Spiegeln	Photo/GeometryTransform (M1404): Drehen um Vielfache von 90 Grad, Spiegeln waagerecht und senkrecht, Zuschneiden (ein Bereich ausserhalb wird benannt abgelehnt, nie still geklemmt), Geraderichten um einen beliebigen Winkel und Perspektivkorrektur. Ein Punkt ausserhalb wird durchsichtig, nie geraten. Offen: das automatische Upright (die Erkennung der Fluchtlinien), und Straighten behaelt die Bildmasse - die Ecken werden abgeschnitten	M
LR8	Effekte: Vignettierung nach Zuschnitt, Koernung	Photo/PhotoEffects (M1404) - radiale Abdunklung mit quadratischer Kennlinie (die Kantenmitte trennt sie von der linearen; Ecke und Mitte waeren bei beiden gleich) und achromatisches Filmkorn (alle drei Kanaele aendern sich um denselben Betrag)	S

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

Punkt	Stand	Was noch fehlt / warum
PDF/UA-Semantik (Leseordnung/Alt-Text inhaltlich)	x menschlich	veraPDF prueft das Maschinell-Entscheidbare; ob ein Alt-Text die Abbildung inhaltlich beschreibt, bleibt menschliche Sichtpruefung.

INF · Betrieb und Auslieferung (Nutzerwunsch)

Wie die Bausteine als Container zusammenkommen. Seit M1412 ist der Wunsch umgesetzt: drei Dienste — Pruefstand, Noten und Druck-Spooler — laufen je als eigener Container, jeder ist dreifach erreichbar (allein und in beiden Standard-Verbuenden), und alle drei kommen aus einem Bild. Dazu kam als Unterbau das Werkzeugbild, das die Fremdwerkzeuge traegt.

#	Punkt	Stand
INF1	Noten als eigener Dienst — alles rund um Noten (Notensatz, MusicXML/MIDI, OMR, Noten-Editor) laeuft als eigene Instanz, auf die zugegriffen wird	M1412 (Playground/Services/DienstProfil + KUONEN_PDFTOOLS_DIENST=musik): eigener Container, eigene Adresse, eigene Oberflaeche - aber dasselbe Bild, dieselbe Entscheidung wie beim Spooler. Browser-belegt in beide Richtungen (mit der Angabe: / -> /noteneditor, genau ein Reiter; ohne sie: 25 Reiter). Ein unbekannter Dienstname faellt auf den vollen Pruefstand zurueck und wird benannt. Ehrliche Grenze: der Dienst schaltet die Oberflaeche um, nicht die Bibliothek - wer eine andere Adresse eintippt, bekommt sie (Gliederung, kein Zugangsschutz)
INF2	Der Noten-Dienst in allen drei Compose-Dateien: docker-compose.yml, docker-compose.tools.yml und eigenstaendig als docker-compose.musik.yml	M1412 - alle drei; ein Waechter haelt sie zusammen (DienstVerbundTests)
INF3	Derselbe Dreiklang fuer den Spooler: allein (docker-compose.spooler.yml, besteht seit M1400) und zusaetzlich in den beiden Standard-Dateien	M1412 - in beiden Standard-Verbuenden; der Pruefstand ist dort Kunde des eigenstaendigen Spoolers (KUONEN_PDFTOOLS_SPOOLER_URL), startet also keinen zweiten. Als einziger Dienst mit veroeffentlichtem Host-Port (631, dem genormten IPP-Port) - ein Drucker, den kein Geraet erreicht, ist keiner
INF4	Auslieferung ueber GitHub nur mit docker-compose.yml und docker-compose.tools.yml	M1412 - so ausgelegt (und in docker/README.md als Auslegung benannt): die uebrigen Compose-Dateien bleiben im Repo, aber weder rebuild.sh noch die Veroeffentlichungs-Kette ruehren sie an. Die zwei ausgelieferten tragen jetzt alle drei Dienste
INF6	Der Waechter zieht mit: TesseractImageTests haelt heute fest, dass es genau zwei Bilder gibt — kommen eigene Dienste dazu, muss er das mit Begruendung abbilden statt umgangen zu werden	M1412 - es bleibt bei zwei Anwendungsbildern: die neuen Dienste sind Container desselben Bildes, kein drittes. Dazu kommt das Werkzeugbild als Unterbau, und der Waechter belegt die Begruendung statt sie zu behaupten: es traegt keine Zeile Anwendungscode (kein ENTRYPOINT, kein COPY --from=build)
INF7	Der Bau fuellt den Cache-Speicher, statt ihn zu nutzen (vom Nutzer gemeldet: 9,22 von 10 GB, ueber hundert Eintraege)	M1412 - Ursache gemessen: der GitHub-Cache ist nach Referenz abgeschottet, der Deploy laeuft aber auf einen Tag - jeder Tag eine eigene Referenz. Was ein Tag-Bau schrieb, konnte der naechste nie lesen: Schreibmuell, der unter dem 10-GB-Deckel ausgerechnet den einzigen brauchbaren Cache (den von main) verdraengte. Dagegen drei Dinge: die teuren Fremdwerkzeuge liegen jetzt als eigenes Bild auf Docker Hub (kein Deckel, keine Verdraengung, gebaut nur bei neuem Schluessel aus tool-versions.env + Rezept, mit der .NET-Fassung im Namen); ein Tag-Bau schreibt keinen Cache mehr (er liest den von main); und ein Aufraeum-Workflow loescht unlesbare Eintraege und haelt den Rest unter Budget. Ehrlich offen: die Wirkung zeigt sich erst am naechsten Bau - belegt ist die Ursache, nicht der Erfolg

#	Punkt	Stand
INF8	Der Docker-Bau brach ab (vom Nutzer gemeldet: dotnet publish Exit 1)	M1412 - .dockerignore schloss docs/ aus und holte eine Datei zurueck; M1411 hatte die Roadmap in fuenf Teildateien aufgeteilt und eingebettet. Im Bau-Kontext fehlten sie, lokal faellt das nie auf. Am echten Docker belegt: mit der Korrektur 6 Dateien im Kontext, ohne sie 1. Ein Waechter haelt csproj und .dockerignore jetzt zusammen - dieselbe Fehlerklasse wie IMG1/IMG22/IMG27

Warum „eigene Instanz“ nicht automatisch „eigenes Bild“ heisst: der Spooler ist seit M1400 ein zweiter Einstiegspunkt desselben Bildes — zwei Container, ein Bild. Das war die kleinere Aenderung und haelt beide Dienste zwangslaeufig auf demselben Quellstand. Fuer den Noten-Dienst ist dieselbe Frage bewusst zu entscheiden, nicht nebenbei.

IMG7 · HEIF/AVIF schreiben — abgeschlossen mit M1496

#	Punkt	Stand	
IMG7 11/11	HEIF/AVIF schreiben (HEVC- bzw. AV1-Intra-Encoder)	Untertabelle unten; IMG7.1–7.10a siehe weiter unten im Archiv	XL

IMG7 · HEIF/AVIF schreiben — 11 von 11 (die letzten beiden Scheiben)

#	Scheibe	Stand
IMG7.11	AVIF schreiben	M1496 - eigener AV1-Intra-Codierer, der denselben Syntaxweg laeuft wie der Dekodierer: an jeder Lesestelle nennt er den gewuenschten Wert, ein schreibender Symbolstrom (Av1SymbolEncoder, Bereichscodierer samt Endbit nach exit_symbol) schreibt ihn - Vorhersage, Entquantisierung, Ruecktransformation, Kontexte und Tabellen sind Zeile fuer Zeile die des Lesers. Werkzeuge: Aufteilung 64x64 bis 4x4 (ungeteilt/geviertelt nach Hadamard-Fehler plus Bits), alle 13 Intra-Modi, Chroma-aus-Luma, DCT/ADST (Vorwaerts-Matrix aus der normativen Rueckrichtung gewonnen), Totzonen-Quantisierung, Kantenfilter und CDEF mit gegen die Quelle gewaehlten Staerken; verlustfrei Walsh-Hadamard und GBR (Matrix 0). Behaelter ueber die vorhandene HEIF-Schreibseite (av01/av1C/ispe/pixi/colr, Deckkraft als einfarbiges AV1-Hilfsbild mit der AVIF-URN, die der Leser jetzt auch liest). Belegt: dav1d (PyAV) dekodiert 12 eigene Stroeme (verlustfrei, 4:2:0/4:2:2/4:4:4/einfarbig, 8/10/12 Bit) Bildpunkt fuer Bildpunkt zu denselben Ebenen wie der eigene Leser; libavif (Pillow) liest verlustfrei samt Deckkraft bit-genau. Bildwandlung, MCP convert_image und Pruefstand /foto schreiben AVIF
IMG7.10b	10 Bit schreiben und lesen	M1496 - neues Bildmodell Rgba64Image (16 Bit je Kanal) neben RgbaImage; AV1 schreibt und liest 10/12 Bit, HEVC schreibt Main 10 / Main 4:2:2 10 / Main 4:4:4 10 und liest 10 Bit (QpBdOffset, Kantenfilter-Skalierung, SAO-cMax, Profil, hvcC/pixi). HeifImageInfo.HighBitDepthPixels traegt die 16-Bit-Pixel, auch fuer Kachelgitter. Belegt: ffmpeg dekodiert die eigenen 10-Bit-HEVC-Stroeme zu denselben Ebenen; alle 8-Bit-Stroeme sind byte-gleich geblieben

IMG6 · AVIF lesen (AV1-Intra-Decoder) — abgeschlossen mit M1413

#	Punkt	Stand	
IMG6 8/8	AVIF lesen (AV1-Intra-Decoder) — in Scheiben wie der HEVC-Decoder (M1111–M1147)	Untertabelle unten: IMG6.1–6.8	XL

IMG6 · AVIF lesen (AV1-Intra-Decoder) — 8 von 8

#	Scheibe	Stand
IMG6.8	Nachfilter (Deblocking, CDEF, Loop Restoration) + YUV->RGB, PixelsDecoded = true	M1413 - alle drei in Normreihenfolge zwischen Rekonstruktion und RGB, samt get_source_sample (§7.17.6: im Streifen die Werte nach CDEF, darueber/darunter davor) und den drei noetigen Bildfassungen. Gegen dav1d auf den rohen Farbebenen: 17 Dateien, 0 Abweichungen - darunter eine, die vorher 73223 von 73728 Bildpunkten daneben lag. Dabei fielen zwei still weggeworfene Rahmenkopf-Angaben auf (loop_filter_delta_enabled/ref_deltas) und ein latenter int-Ueberlauf in Av1TileGroup.Split. Filmkorn (§7.18.3) bleibt bewusst draussen: Absicht-Rauschen, kein Bildinhalt

v4-LPP · PowerPoint — in M1420 geschlossen

#	PowerPoint-Funktion	Stand	Aufw.
LPP1	Layouts + Master + Platzhalter (sldLayout/sldMaster/ph)	M1420: Lesen (Vererbung sldMaster -> sldLayout -> Folie) UND Schreiben - eine Form, die ihre Lage vom Platzhalter erbt, wird auch wieder ohne eigenes <a:xfrm> geschrieben; ein Schreiber, der die Lage einbetoniert, kaeme durch einen blossen Positionsvergleich durch. Noch nicht gebaut: die Platzhalter aller Layouts landen in einem geschriebenen Layout	L
LPP3	Formen-Geometrie (prstGeom) + Fuellung/Kontur/Verlauf/Schatten	M1420: Lesen war vollstaendig, Schreiben jetzt ebenso - jede bekannte Geometrie (Ellipse, RoundRect, vier Pfeile, Stern, Callout), gradFill, Kontur mit Breite, effectLst/outerShdw	L
LPP4	Aufzaehlungen + Absatzformat (buChar/buAutoNum, Einzuege, Zeilenabstand, Tabulatoren)	M1420: gelesen UND geschrieben (lvl, marL, lnSpc, tabLst in der OOXML-Abfolge); die Tabulator-Sprungpositionen werden jetzt wirklich angewandt (Vorgabe ein Zoll, wo keine festgelegt sind). Noch nicht gebaut: die Ausrichtung einer Tabulator-Position (algn)	M

v4-MUS · Notensatz und Noten-Editor — in M1421 geschlossen (M1426: MUS2 gebuendelt, MUS3 feine Stichregeln)

#	Punkt	Stand	Aufw.
MUS2	Notenschrift anbinden und buendeln: SMuFL-Glyphennamen -> Codepunkte -> Glyph-Nummern, Referenzschrift Bravura (SIL OFL 1.1) in Core ab Werk gebuendelt	M1426: Music/BundledMusicFont + SmuflMusicFont.Bundled. Offizielle Steinberg Bravura.otf als eingebettete Ressource in Kuonen.PDFTools.Core gebuendelt. MusicPdf.Render, ScoreEngraver, MCP und Noten-Editor nutzen Bravura ab Werk als Vorgabe - Schluessel (Violin/Bass/Bratsche), Vorzeichen, Pausen, Notenkoepfe und Faehnchen werden ohne externe Schriftanforderung vollstaendig und sauber gesetzt. Gezielte Vektor-Unterdrueckung ueber SuppressMusicFont moeglich	S

#	Punkt	Stand	Aufw.
MUS3	Die feinen Stichregeln: Balkenneigung, Balkenraster (Sit/Straddle/Hang), Ausweichen bei Kollisionen (Bindeboegen, Fermaten, Beischriften), elastischer optischer Abstand	M1426: EngravingRules + ScoreEngraver + ScoreEngraver.Expression. Melodische Konturanalyse fuer waagerechte Balken bei Wiederholungen und Wechselnoten, Einrasten von Balkenneigung an Notenlinien (Sit, Straddle, Hang), Halsverlaengerung bis zur Mittellinie fuer Noten auf Hilfslinien, einheitliche Halsseite in Balkengruppen, dynamisches vertikales Ausweichen von Beischriften (Dynamik, Gabeln, Silben, Akkorde) und Fermaten, dynamische Woelbungsanpassung von Bindeboegen zur Umgehung dazwischenliegender Notenkoepfe, elastische Spaltenbreitenkompensation fuer gestaffelte Vorzeichen und lange Silben	XL
MUS5	Alles am Notentext: Dynamik, Artikulation, Bindeboegen, Triller/Verzierungen, Oktavierungen, Pedal, Fermaten, Atemzeichen, Tempo- und Vortragsangaben	M1421: Music/MusicExpression + ScoreEngraver.Expression - je ins Modell, in den Notenstich und ins PDF. Die Stichregeln sind eingehalten: Artikulation auf der halsabgewandten Seite, der Bindebogen woelbt sich entgegen der Halsrichtung, und er ist strukturell vom Haltebogen getrennt (ein Haltebogen verbindet dieselbe Tonhoehe, ein Bindebogen verschiedene)	L
MUS6	Text an den Noten: Gesangstext (mehrere Strophen, Silbentrennung, Melismen), Akkordsymbole, Fingersatz, Generalbass	M1421. Ein stiller Fehler dabei behoben: DrawTexts legte sich einen eigenen Bericht an und warf ihn weg - ein nicht setzbares Zeichen in einer Silbe wurde brav bereinigt und die Meldung darueber an niemanden gerichtet; der Aufrufer sah einen stillschweigend gekuerzten Gesangstext	M

#	Punkt	Stand	Aufw.
MUS7	Mehrere Systeme: Klavier-Akkolade, Chor-/Orchesterpartitur, Systemgruppen und Klammern, Stimmenauszug aus der Partitur, Transposition (klingend notiert)	Geschweifte Akkolade und eckige Systemklammer ueber EngravingRules.Groups (M1404), 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. Part.StaffCount > 1 (Klavier/Orgel brauchen zwei Notenzeilen fuer EINE Stimme): Part.ExtraStaves traegt die Takte je zusaetzlicher Notenzeile, Instrument.StaffClefs ihren eigenen Vorgabeschluessel; der Notenstich zerlegt jede Stimme in ihre Notenzeilen und zeichnet zwischen den EIGENEN Notenzeilen eines Instruments automatisch Akkolade und durchgehende Taktstriche (unabhaengig von EngravingRules.Groups - es ist EIN Instrument, keine mehreren Stimmen), nur die oberste traegt den Namen; die Umfangspruefung prueft jetzt alle Notenzeilen. Der Stimmenauszug (Music/PartExtractor.Extract, nach Index oder Name): notiert (Vorgabe, das Modell haelt je Stimme ohnehin die geschriebene Tonhoehe) oder mit concertPitch: true klingend umgerechnet (Transposition.ToSounding/KeyToSounding, danach transponiert der Auszug nicht mehr - kein doppeltes Transponieren bei erneuter Extraktion). Zusammenhaengende leere Takte fasst PartExtractor.CollapseRests automatisch zu einer mehrtaktigen Pause zusammen (MUS8-Ausschnitt, zwingend fuer einen brauchbaren Auszug) - ein Takt mit einer Note ODER einem eigenen Schluessel-/Tonart-/Taktart-/Tempowechsel bleibt eigener Anker, damit nichts verloren geht; die gedruckten Taktnummern bleiben dabei die echten der Partitur. MusicXML liest UND schreibt mehrere Notenzeilen je Stimme (<staves>/<staff>/<clef number>; ohne zusaetzliche Notenzeile byte-identisch zu vorher). Ehrlich offen: der Noten-Editor (/noteneditor) setzt weiterhin nur EINE Notenzeile je Spur (der Notenstich kann mehr, die Spur-Klasse nutzt es noch nicht); Tonart/Taktart gelten in MusicXML fuer alle Notenzeilen einer Stimmgruppe gleich (ein je Notenzeile eigenes <key number> wird nicht gelesen)	L
MUS13.6	Freistehend wie der grosse Editor: Werkzeugleiste, Eigenschaften-Feld, Rueckgaengig-Kette, Speichern/Laden im Arbeitsbereich	M1421 - MusicEditorState haelt Dokument und Verlauf als reine Funktionen (jede Aenderung ergibt ein neues Dokument), gespeichert wird ueber den vorhandenen MusicXml-Weg im Arbeitsbereich	
MUS13.8	Auswahl und Ziehen: einzeln, Rahmenauswahl, mehrere; verschieben in Tonhoehe UND Zeit	M1421 - MusicSelection nach dem Vorbild von EditorMarquee, Verschieben ueber MusicCursorEditing. Die Anker kommen aus der gesetzten Partitur, damit es nur eine Takteinteilung gibt	
MUS13.11	Klaviatur unten, ein- und ausblendbar; eine Taste setzt die Note an der aktuellen Stelle	- die Klaviatur laesst sich ein- und ausblenden, eine schwarze Taste ist ein erhoehter Ton (ihr Vorzeichen schlaegt das der Palette); MusicKeymap.TonhoeheAusMidi rechnet eine MIDI-Nummer in eine Tonhoehe, die Wahl cis/des trifft der Aufrufer statt zu raten	
MUS13.12	Eingabe ueber die Rechnertastatur (Tonhoehe, Notenwert, Vorzeichen, Weiterruecken)	M1421 - a-g setzen, die Pfeiltasten verschieben die Auswahl in Tonhoehe und Zeit, Entf loescht; browser-verifiziert am laufenden Pruefstand	

#	Punkt	Stand	Aufw.
MUS13.10a	Noten verbinden von Hand: Haltebogen, Bindebogen und Balken je Eintrag setzen und loesen	M1488 (KMU20) - Auswahlfelder 'Haltebogen' und 'Balken' je Eintrag, 'Bogen' in der Werkzeugleiste	
MUS13.14	Dynamik, Artikulation, Gesangstext und Akkordsymbole im Editor bedienen	M1488 (KMU20) - Dynamik und Artikulation als Auswahl in der Werkzeugleiste, Silbe und Akkordsymbol als Feld je Note. Grenze, im Editor selbst benannt: je Note eine Silbe der ersten Strophe	
MUS13.15a	Mehrere Notenzeilen je Spur: beide Zeilen sichtbar, Treffer-Ermittlung je Zeile, Wechsel ueber Knopfleiste oder Klick	M1488 (KMU19)	
MUS13.16	Akkorde eingeben: mehrere Koepfe auf derselben Zeitstelle	M1488 (KMU19) - ueber den Umschalter 'Akkord' in der Werkzeugleiste (der naechste Ton kommt an die letzte Note), samt 'Akkord aufloesen'; nicht ueber Umschalt+Klick	
MUS13.17	Mehrere Stimmen im selben Takt, Halsrichtung nach Stimmenrolle	M1488 (KMU19) - zwei Stimmen je Notenzeile ueber die Knopfleiste 'Stimme'; mehr als zwei sind bewusst nicht vorgesehen (Begruendung in KMU29)	
MUS13.18	Bindebogen, Haltebogen, Dynamik, Artikulation, Tempoangabe im Editor setzen	M1488 (KMU20) - deckt sich mit 13.10a und 13.14; Tempo als Zahl und als gedruckte Angabe je Eintrag	
MUS13.19a	Schluessel- und Tonartwechsel mittendrin	M1488 (KMU20) - je Takt ueber die Taktangaben des Eintrags	

#	Punkt	Stand	Aufw.
MUS16	Notenblatt-Erkennung (OMR): aus dem Bild eines gedruckten Notenblatts zurueck zum Score	M1494 (vorher M1404-M1491 als): Modul MusicOcr. Am echten Scan stimmt die Takteinteilung: zwei Stimmen (Melodie, Begleitung) ueber drei Systeme mit je Auftakt und 16 Takten, nummeriert 0-16 (vorher 54 -> 29 -> 26 Takte ueber sechs unabhaengige Stimmen); Taktart 4/4, Tonart und Schluessel gelesen, die Melodie Note fuer Note richtig (eigener Test gegen die gedruckten Noten). Die groesste Abweichung hatte fuenf Ursachen, je mit eigenem Test: ein schwarzer Fleck aus grauem Papier in der Seitenmitte (Median-Grund wird vor der Hysterese abgezogen), dickere, versetzte Linien dort (Linienentfernung nach dem gemessenen Verlauf der Linie statt einer Staerke je Zeile - nebenbei bleiben offene Koepfe im Zwischenraum ganz und vier B werden gelesen), Systeme (Systemstrich/durchgehender Taktstrich; Stimmen laufen ueber Zeilen und Seiten weiter), Taktstriche, die eine beruehrende Klammer zum Hals machte (Hals nur, wenn am Strichende ein Kopf ansetzt; Kernlaenge per Median; Doppelstriche zaehlen einmal) und die Taktangabe, deren Vieren einen Strich ergaben. Neu gebaut: Taktart (TimeSignatureReader: Ziffern ueber Vorlagen aus Bravura, roh und geglaettet verglichen, C und Allabreve geometrisch, Wechsel im Lauf der Zeile, unsichere Lesung nur durch die sichere derselben Systemstelle bestaetigt), Auftakt (kurzer erster Takt, den der letzte ergaenzt) und Taktpause, Mehrtaktpausen samt Zahl (MultiRestReader, Vorlagen aus zwei Schriften), Akkorde (Klumpen in Koepfe zerlegt - Terzen/Quarten als Kette, Sekunden in zwei Spalten, gefuellt im Bild mit Linien; Koepfe an einem Hals sind eine Note mit mehreren Tonhoehen), zwei Stimmen einer Zeile nach der Halsrichtung, Halte- und Bindeboegen (ArcReader, Stuecke auf Linien werden verbunden; zwei halbe Boegen ueber den Zeilen- oder Seitenumbruch werden beim Zusammensetzen der Stimme zu einem), die Hilfslinien-Pflicht fuer Koepfe ausserhalb der Zeile (Akkordsymbole kamen als ganze Noten), Halsstummel, Loch im Graubild, Balken nur am Hals, Punkt nur frei stehend, Sperrbereiche beim Pausenpunkt jetzt durch einen Test belegt (ohne sie blieb vorher jeder Test gruen). Grundsatz gehalten: nichts Gelesenes ist falsch, Unsicheres wird weggelassen und gemeldet. Jede Scheibe mit Sabotage-Probe. Fremd-Orakel: oemer (BSD-2, Wegwerf-venv) neu befragt: die sechs eigenen Ausschnitte der neuen Scheiben scheitern schon in seiner Linienfindung (keine zweite Meinung zu Akkorden, Boegen, Mehrtaktpausen, Taktart, zwei Stimmen); am echten Scan liest es eine Stimme mit 32 Takten, ein Kreuz, keine Taktart, und die Melodie stimmt in 13 von 17 Takten unabhaengig mit den gedruckten Noten - dort liest die eigene Erkennung dasselbe (Befund in MusicOcr/Vorlagen/HERKUNFT.md). Ehrlich benannte Grenzen (keine offenen Punkte, sondern die Reichweite der Geometrie): die Begleitstimme des Scans ist unvollstaendig (farbige Achtelakkorde in Balkengruppen fehlen teils - weggelassen, nicht falsch); mehr als zwei Stimmen in einer Zeile trennt der Notenstich nur durch Farbe; Taktart- und Mehrtaktziffern einer Schrift, die keiner der zwei Vorlagenschriften aehnelt, bleiben ungelesen und gemeldet; Handschrift braucht ein gelerntes Modell	XL

v4-IMG · Nachtrag — zwei Zeilen, die in der Restliste haengengeblieben waren

Gefunden in M1422 auf Nutzerfrage. Beide waren gebaut; die alte Statusregel konnte sie nur nicht sehen (bei IMG22 ueberstimmte ein der Beschreibung das der Statusspalte, bei IMG17 war nach M1400 schlicht die Marke nicht nachgezogen).

#	Punkt	Stand	Aufw.
IMG17	JPEG 2000 verlustbehaftet schreiben: 9/7-Wavelet + Quantisierung + Ratensteuerung (welche Codierdurchlaeufe fallen weg — der PCRD-Optimierer). Alles Uebrige steht seit M1306	M1344: 9/7-Wavelet + Quantisierung je Teilband (Exponent/Mantisse im QCD), Stufe 1..100, auch in convert_image (lossy); openjpeg liest es mit hoechstens 3 Abweichung. + M1400: die Ratensteuerung (PCRD-opt) - eine echte Zielgroesse in Bytes bzw. ein Kompressionsverhaeltnis, ueber die konvexe Huelle im Rate-Distortion-Raum und einen Lagrange-Multiplikator. Ueber ImageConvertOptions und convert_image (target_bytes/compression_ratio) erreichbar	L
IMG22	.j2k im Erkennungsweg - Fehler derselben Klasse wie IMG1: RasterImage.Parse kannte JPEG 2000 nicht, obwohl Decoder und Encoder da waren	M1307: Codestream (FF4F FF51) und JP2-Box erkannt; convert_image/images_to_pdf/Dokument-Import nehmen .j2k jetzt als Quelle	S

v4-ARC · Archive packen und entpacken — in M1433 geschlossen (alle Punkte)

#	Format / Funktion	Lesen	Schreiben	Aufw.
ARC1	ZIP vollstaendig: ZIP64 (>4 GB, >65535 Eintraege), AES-Verschluesselung (WinZip AE-1/AE-2) und das alte ZipCrypto, weitere Methoden im ZIP (BZIP2, LZMA, Zstd), Kommentare, Datei-Attribute und Zeitstempel	M1426: ZIP64 (>4 GB, >65535 Eintraege), WinZip AES (AE-1/AE-2, 128/192/256 Bit), ZipCrypto, Deflate/Store/Bzip2/LZMA/Zstd, Kommentare, Unix-Rechte, erweiterte Zeitstempel (0x5455/0x000A)	M1426: ZIP64-EOCD/Locator, WinZip AES-128/192/256, ZipCrypto, Deflate/Store/LZMA, Unix-Rechte (0644/0755), erweiterte Zeitstempel (0x5455), Kommentare	M
ARC2	BZIP2 (.bz2) — Burrows-Wheeler, Huffman	M1423: eigener Dekodierer (Huffman mit Tabellenwechsel, MTF, inverse BWT, RLE), byte-genau gegen Pythons bz2	der Kodierer ist ein eigenes Stueck Arbeit (Sortierung aller zyklischen Verschiebungen)	M
ARC7	TAR-Ketten: .tar.gz, .tar.bz2, .tar.xz, .tar.zst als EINE Einheit behandeln, dazu die Unix-Rechte/Besitzer/Symlinks	M1427: .tar.gz, .tar.bz2, .tar.xz und .tar.zst als EINE Einheit erkannt und gelesen; Unix-Rechte (0755/0644/0600), Besitzer/Gruppen, UID/GID und Symlinks in ArchiveEntry und TarArchiveWriter	.tar.gz und .tar mit konfigurierbaren Unix-Rechten und Besitzern geschrieben (TarWriteOptions)	S
ARC3a	XZ / LZMA / LZMA2 lesen (.xz, .lzma)	M1426: XzDecoder, Lzma2Decoder und LzmaCodec.Entpacke samt Bereichsdekodierer, Fenster und Modell	nur LZMA (LzmaCodec.Packe, LzmaRangeEncoder) - LZMA2 und XZ schreiben ist ARC3b	L
ARC5a	Zstandard entpacken (.zst)	M1426: ZstdDecoder mit FSE- und Huffman-Tabellen	— der Packer ist ARC5b	L
ARC6	LZ4 (.lz4) — schnell, in Linux-Werkzeugketten verbreitet	M1426: Lz4BlockCodec und Lz4FrameCodec, Rahmen-Pruefsumme ueber die eigenen XxHash32/XxHash64	M1426: Block- und Rahmenformat, die Block-Endregeln je einzeln belegt	S

#	Format / Funktion	Lesen	Schreiben	Aufw.
ARC12a	compress (.Z) — das alte Unix-Format	M1426: CompressLzwCodec (LZW mit wachsender Codebreite, Blockmodus)	M1426: CompressLzwCodec.Encode	M
ARC3b	LZMA2 und XZ schreiben	siehe ARC3a	M1433: Lzma2Encoder (Kontrollbyte, gepackte/ungepackte Bloecke bis 64 KiB; Woerterbuch-Reset und Eigenschaften-Uebermittlung als ZWEI Fahnen - ein frueher ungepackter Block liess die Eigenschaften sonst ungesagt) und XzEncoder (Stream-Kopf/-Fuss, Index, CRC32/CRC64/SHA-256)	M
ARC4	7z (.7z): LZMA/LZMA2, BCJ, feste Blockbildung, Verschluesselung	M1433: Signaturkopf, gepackter Hauptkopf, Verfahrensketten Speichern/LZMA/LZMA2/BCJ-x86/Delta/AES-256 (die Reihenfolge aus den Bindepaaren, nicht aus der Kopf-Reihenfolge - py7zr schreibt sie je nach Kette verschieden), solide Bloecke; gegen py7zr und 7-Zip	M1433: Speichern/LZMA/LZMA2, BCJ-x86, solide, gepackter Kopf; AES-Schreiben nicht	L
ARC5b	Zstandard packen	siehe ARC5a	M1433: Rahmen mit XXH64, Bloecke roh/RLE/gepackt, Huffman-Literale, drei FSE-Tabellen, Hash-Kette ueber Blockgrenzen - und DREI Fehler im Leser, die erst zstd zeigte (FSE-Huffman-Gewichte um eins zu viel, kanonische Codes falsch sortiert)	L
ARC8	RAR entpacken (RAR4 und RAR5)	M1433: beide Behaelter, RAR-2.9- und RAR-5.0-Verfahren samt Filtern (Standard-VM-Filter am Fingerabdruck erkannt); RAR5-Codelaengen absolut statt als Unterschied (an einem echten Mehrblock-Archiv gefunden). Nicht: mehrteilige Archive, Wiederherstellungsdaten, verschluesselte RAR	nie — die RAR-Lizenz untersagt einen Kompressor; rechtliche, keine technische Grenze	L
ARC9	CAB (MSZIP/LZX)	M1433: Store, MSZIP und ein eigener LZX-Dekodierer; der Ordner-Strom laeuft ueber Dateien UND CFDATA-Bloecke hinweg und wird je Ordner einmal ausgepackt (wer je Block neu beginnt, liefert ab Block 2 Muell)	M1433: Store und MSZIP (kein LZX-Packer)	M
ARC10	Apple: .dmg (HFS+/APFS) und Apple Archive (.aar/.yaa)	M1433: UDIF mit Stueckarten roh/Null/ADC/zlib/bzip2/LZFSE/LZMA, Partitionssuche (direkt/GPT/Apple Partition Map), HFS+/HFSX samt Links und decmpfs, APFS (Pruefpunkt, Objektkarten, Fletcher-64, decmpfs mit zlib/LZVN/LZFSE), eigener LZFSE/LZVN-Entpacker, Apple Archive blank und im pbz*-Mantel; gegen libmodi, libfshfs, libfsapfs, liblzfse/libfmos. Nicht: verschluesselte/versiegelte/Fusion-APFS, LZBITMAP, bvx1, Pruefsummen im .dmg, das HFS-Journal	nicht gebaut	L

#	Format / Funktion	Lesen	Schreiben	Aufw.
ARC11	ISO 9660 / UDF	M1433: ISO mit Joliet und Rock Ridge (CE-Fortsetzungen, rr_moved, mehrteilige Dateien), UDF 1.02-2.60 mit allen vier Partitionsarten (physisch, Sparable, VAT, Metadaten-Partition); gegen xorriso/genisoimage/mkudffs, bsdtar und den Linux-Kern. Nicht: verschraenkte Dateien, Mehrsitzung, ZF, benannte Stroeme	M1433: ISO 9660 mit Joliet und Rock Ridge, byte-gleich wiederholbar, von bsdtar/xorriso/isoinfo/7z/Kern gelesen (Fortsetzungsflaechen direkt hinter dem Verzeichnis, sonst verliert libarchive lange Namen). Nicht: UDF schreiben, El Torito, Dateien ueber 4 GiB	M
ARC12b	StuffIt, cpio, ar	M1433: cpio (newc/crc/odc/binaer, Hardlinks), ar (GNU/BSD, Symboltabelle, .deb), StuffIt klassisch und 5 mit den Methoden 0/1/2/3/13/15; gegen GNU cpio/ar, bsdtar, dpkg-deb, unar/lsar. Nicht: StuffIt-Verschluesselung, Methoden 5/6/8/14, StuffIt X	M1433: cpio (newc) und ar; StuffIt nicht (kein freier Packer zum Gegenpruefen)	M
ARC13	Fassade + MCP + Pruefstand	M1423/M1433: ArchiveDetector/ArchiveReader/ArchiveWriter ueber ALLE Formate, archive_list/archive_extract/archive_create (Gruppe archive), der Reiter /archive	M1433: archive_create und der Reiter packen zip/tar/Ketten/7z/cab/cpio/ar/iso und die Ein-Datei-Mantel	S

Ehrlich zu RAR: entpacken ist nachgebaut; packen nicht und nie — die RAR-Lizenz untersagt, aus den veroeffentlichten Unterlagen einen RAR-Kompressor zu bauen. Allen Formaten gemeinsam: das Archiv liegt ganz im Speicher (fuer sehr grosse Abbilder braeuchte es einen stroemenden Weg).

v4-LIC · Lizenz-Manager — in M1427 geschlossen (7 von 7)

*„ich moechte das der lizenz maenager endlich detailiert ist aber gruppen oder einzel auswahl

moeglich ist"*

Was heute da ist, am Code nachgesehen. Licensing/LicensedFeatures kennt neun grobe Merkmale — gpu.acceleration, batch.processing, commercial.use, security.removal, pdf.creation, instance.full, show.downloads, hide.extras, signature.export — plus license.unlimited. Eine Lizenz schaltet damit ganze Themenbloecke frei oder nicht; ein einzelnes Werkzeug laesst sich gar nicht adressieren.

Der tragende Befund: die halbe Arbeit ist schon getan. Die Einteilung existiert (Mcp/McpToolGroups: 13 Themen ueber 192 Werkzeuge — genau die Gliederung, die der Pruefstand unter „Werkzeuge im Ueberblick“ zeigt), und McpToolListing.ForList filtert bereits danach (M1244/M1267: KUONEN_PDFTOOLS_BLOCKED_GROUPS nimmt gesperrte Gruppen aus tools/list UND aus dem Aufruf). Was fehlt, ist nicht der Filter, sondern die Quelle der Entscheidung: heute ist es der Schalter des Betreibers, kuenftig zusaetzlich das Recht aus der Lizenz. Es darf kein zweiter Filter entstehen — sonst hat die Anwendung zwei Stellen, an denen ein Werkzeug verschwinden kann, und niemand findet mehr heraus, welche gegriffen hat.

#	Punkt	Stand	Aufw.
LIC1	Das Lizenzmodell wird fein: neben den neun groben Merkmalen traegt eine Lizenz eine Liste freigegebener Gruppen (AllowedGroups) und eine Liste einzelner Werkzeuge (AllowedTools). Abwaertskompatibel: eine Lizenz ohne diese Listen wirkt wie bisher — sonst faellt beim ersten Einspielen jedem Bestandskunden die Haelfte weg		M
LIC2	Eine einzige Stelle entscheidet — die Lizenz wirkt durch denselben McpToolListing.ForList/IsBlocked-Weg (McpToolPolicy.CheckAccess) wie die Sperrliste, nicht daneben. Die Entscheidung liefert dabei den Grund mit (Allowed, NotLicensed, Blocked, Unknown), damit alle Faelle praezise unterscheidbar bleiben		M
LIC3	Der Lizenz-Manager als Oberflaeche — in der Darstellung der Werkzeug-Uebersicht: 13 Gruppen mit Zaehler, aufklappbar, Haken auf der Gruppe ODER auf dem einzelnen Werkzeug, „alle/keine“ je Gruppe, Suche ueber alle 204. Eine halb angehakte Gruppe muss als solche zu sehen sein	M1427: /lizenzmanager - 17 Gruppen mit Zaehler, aufklappbar, Haken auf Gruppe ODER Einzelwerkzeug, Suche ueber alle 204; die halb angehakte Gruppe ist als solche zu sehen (orange, halbgefuelltes Quadrat). Rechenkern Playground/Model/LizenzManager (reine Funktionen). Der Manager stellt keine Lizenz aus - er gibt die Mengen zum Kopieren aus	L
LIC4	Dieselbe Auswahl beim Ausstellen — LicenseGen und LicenseAdmin schneiden eine Lizenz mit genau derselben Auswahl zu, statt nur die groben Merkmale zu kennen (--groups/--tools im CLI bereits aktiv)	M1427: geteilte Pruefung Licensing/LicenseSelectionCheck im Kern - beide Aussteller rufen sie; ein unbekannter Name wird benannt abgelehnt (Rueckgabewert 5, keine Datei), dazu --blocked (war fest auf null verdrahtet) und die Auflistung dessen, was die Lizenz wirklich freigibt. --features bleibt ungeprueft (Kennungen sind bewusst offen)	M
LIC5	Sichtbar statt still — ein nicht lizenziertes Werkzeug verschwindet aus tools/list und nennt beim Aufruf den Grund samt dem, was die Lizenz traegt (McpToolNotLicensedException meldet Werkzeug, Gruppe, Lizenznehmer sowie freigegebene Gruppen und Einzelwerkzeuge). Ein Werkzeug, das wortlos fehlt, schickt den Nutzer auf die falsche Fehlersuche		S
LIC6	Der Pruefstand zieht mit — Reiter und Seiten folgen derselben Lizenz wie die MCP-Werkzeuge; heute haengt die Oberflaeche an den groben Merkmalen (show.downloads, hide.extras)	M1427: Playground/Model/SeitenLizenz + PlaygroundSecurity.SeiteErlaubt, durchgesetzt an einer Stelle um @Body (eine Schranke je Seite liefe auseinander). Abwaertskompatibel: eine Lizenz ohne Gruppen-/Werkzeugangaben verhaelt sich genau wie bisher, und eine Seite ohne Sachgebiet wird nie gesperrt. Sichtbarkeits-Regel, kein Zugangsschutz - die Faehigkeiten selbst schuetzen die Feature-Gates und die MCP-Durchsetzung	M

#	Punkt	Stand	Aufw.
LIC7	Der Nachweis — eine Lizenz, die genau eine Gruppe traegt, gibt genau deren Werkzeuge frei und kein einziges daneben; dazu der Gegenbeleg (ohne die Gruppe ist keines davon da) und der Fall „Gruppe gesperrt, aber ein Einzelwerkzeug daraus ausdruecklich lizenziert“ (McpToolGroupLicensingTests)		M

Eine Falle, die der Zuschnitt mitbringt und die vorab benannt gehoert. Eine Lizenz, die Werkzeugnamen aufzaehlt, bindet sich an diese Namen: wird ein Werkzeug umbenannt oder aufgeteilt, verliert eine bestehende Lizenz es stillschweigend. Das ist kein Grund, es zu lassen — aber es braucht entweder stabile Kennungen neben dem Anzeigenamen oder eine gepflegte Umbenennungs-Tabelle, und einen Waechter, der beides zusammenhaelt (wie der M559-Drift-Waechter es fuer die Manifeste tut).

Und eine Abwaegung, die zum Modell gehoert. Je feiner eine Lizenz schneidet, desto mehr Stellen gibt es, an denen die Durchsetzung umgangen werden kann. Genau deshalb ist LIC2 kein Beiwerk, sondern der Kern: eine Stelle, die entscheidet, und ein Aufruf, der den Grund nennt.

#	Punkt	Stand	Aufw.
KDX20	PowerPoint-Rest nach KDX5: der Platzhalter-Bezug einer Form (welcher Platzhalter des Layouts sie fuellt) und die Themen-Farbverweise an einzelnen Formen (schemeClr) werden gelesen und geschrieben, erreichen aber das .kdoc nicht — am Code gemessen: Placeholder und schemeClr kommen in src/.../Pptx/ vor, in src/.../DocumentFormat/ nicht. Dazu die Musterseiten-Bindung. Folge: eine Form verliert ueber den .kdoc-Rundlauf ihre Herkunft aus dem Layout und ihre Themenfarbe wird zur festen RGB-Angabe — sie folgt einem Themenwechsel danach nicht mehr	M1475	M
KMU1	Der Noten-Datenteil im .kdoc - der Schluesselstein: Layout/MusicScoreInfo als Bestand und der .kdoc-Teil musicScore (Schema kuenftig §8.30) mit dem heutigen Modell vollstaendig, dazu ein MusicFrame als Ansicht ueber MusicPdf.DrawSystem - dasselbe Muster wie xlsxWorkbook in KDX6a, kein zweiter Notensatz. Neue Architektur-Kante Layout -> Music. Rundlauf-Waechter Score -> .kdoc -> Score. Ohne diese Scheibe haengt jede andere in der Luft	M1476	L
KMU3	MusicXML liest und schreibt, was das Modell traegt. Gemessene Luecke: Liedtext, Dynamik, Gabel, Bogen, Artikulation, Verzierung, Oktavierung, Pedal, Akkordsymbol, Fingersatz, Generalbass, Tempotext, Atemzeichen und mehrtaktige Pause werden weder gelesen noch geschrieben, obwohl das Modell sie seit M1421 traegt. Dazu: barline, print und sound werden in einem case … break still verworfen (MusicXmlReader.cs:353-356) — wenigstens melden. Drei veraltete Klassendoku-Stellen berichtigen	M1476	M
KMU2	Direktimport ohne PDF-Umweg: MusicXmlToKdoc und MidiToKdoc ueber die DocumentConverter-Weiche, KdocToMusicXml zurueck; Fundgrube-Andockung fuer den gemeldeten Verlust	M1476	M
KMU14	Der Editor speichert .kdoc (MusicXML bleibt als Export): MusicEditorSpeicher umstellen, damit die Editor-Einstellungen mitkommen — heute gehen Seitenformat, Rand, Notengroesse und Systemgruppen beim Laden verloren (MusicEditorState.cs:347-350)	M1476	M
KMU15	Der Editor kann, was das Modell kann: MusicEditItem traegt heute sechs Angaben, BaueScore legt je Takt genau eine Voice an und fuellt ExtraStaves nie — ein Klavier bekommt eine Notenzeile statt zwei, obwohl der Stich seit M1457 beides kann. Erweitern um Akkord, Stimme, Notenzeile, Dynamik, Artikulation, Bogen und Liedtext; deckt Roadmap MUS13.14 bis 13.22	M1476	XL

#	Punkt	Stand	Aufw.
KMU5	N-tolen wirklich zeichnen: Klammer und Ziffer setzen, Note.Tuplet zur Liste fuer verschachtelte N-tolen; der Meldecode TupletBracketMissing faellt weg. Heute wirkt das Feld auf Dauer und Spaltenbreite, wird aber nie gezeichnet (ScoreEngraver.cs:951)	M1476	M
KMU4	Wiederholungen und Sprungmarken: Taktstrich-Art links und rechts, Wiederholungsanfang und -ende mit Durchgangszahl, Volte, Segno, Coda, Fine, Da Capo, Dal Segno, Probenbuchstabe, Auftakt; Stich und SMuFL-Namen; im .kdoc und in MusicXML	M1476	L
KMU11	Text und Beiwerk: Textdichter, Arrangeur, Copyright, Widmung am Score; Fussnote in der Partitur; Spielanweisung an beliebiger Zeitstelle statt nur am Taktanfang	M1476	S
KMU10	Layout von Hand: System- und Seitenumbruch von Hand, Abstand je System, Notengroesse je Stimme, versteckte Elemente, gedruckte Taktzahlen (Measure.Number existiert und wird heute nie gedruckt)	M1476	M
KDX21	PowerPoint-Restposten nach KDX20: der Platzhalter-Bezug einer Form steht jetzt im .kdoc, aber PptxPresentationWriter schreibt <p:ph> nur im Text-Zweig — an einem Bild, einer Tabelle oder einem Diagramm geht er beim Zurueckschreiben verloren und wird ueber unsupported gemeldet (ihn dort trotzdem zu setzen waere genau das Feld, das niemand ausliest). Dazu: eine Themenfarbe mit Veraenderern (lumMod/tint/shade) behaelt den festen Hex und wird benannt — der blosse Rollenname ergaebe eine andere Farbe; und die Layout-Platzhalter aller Layouts liegen weiterhin in einer Liste (vorbestehende LPP1-Grenze)	M1477	M
KDX6b	Excel-Rest nach KDX6a: Pivot, Zellanker und Diagramm-Quellbezug (c:f) erreichen das .kdoc nicht — Bilder, Diagramme und Pivot-Tabellen stehen in der Ansicht, nicht im Bestand, und ueberleben den Rueckweg nicht (XLSX_KDOC_IMAGE_VIEW_ONLY/_CHART_VIEW_ONLY/_PIVOT_VIEW_ONLY). Dazu: der XlsxWriter kennt vom Zellformat nur den Fett-Schnitt — Fuellfarbe, Schriftfarbe, Zellrahmen und Ausrichtung stehen im .kdoc, fehlen aber in der geschriebenen Mappe (XLSX_KDOC_CELL_STYLE_DROPPED, es braeuchte eine Schrift-/Fuellungs-/Rahmentabelle neben der gebauten Zahlenformat-Tabelle); Spaltenbreiten, Zeilenhoehen, Blattschutz, Kopf-/Fusszeilen und Makros liest schon XlsxReader nicht; der Originalteile-Speicher (KDX11) ist fuer .xlsx nicht angebunden (XLSX_KDOC_ORIGINAL_PARTS_MISSING); und eine Aenderung an den gesetzten Tabellen der Ansicht wirkt nicht auf die Mappe (es braeuchte jede Zelle als eigenes Objekt MIT Zellbezug im Layout)	M1477	L
KDX19	Stellen in der Export-Sicht (Befund M1460). M1462 hat die drei Folgen weitgehend geschlossen: der Zeichenversatz war nie verloren (er haengt am Anker) - es zerlegte nur niemand die Laeufe daran; FlowDocumentLevel.PlaceMarks tut das jetzt, die Kommentar-Marke sitzt in HTML an der kommentierten Stelle. Die Zuordnung ContainerAbschnitt kannte FlowSectionDocument und warf sie weg - FlowSectionRange.ContainerCount traegt sie (im .kdoc gesichert), daraus je Abschnitt eine benannte @page-Regel in HTML und ein eigenes w:sectPr in .docx. Die Noten scheiterten an der Nummer, nicht am Anker - NumberNotes vergibt eine Reihe fuer Absatz-Fussnoten und Noten der Dokument-Ebene, HTML setzt sie. Offen: .odt-Abschnitte (eigene style:page-layout/style:master-page je Abschnitt - ein eigener Baustein), Docx-/Markdown-/Klartext-Noten, die Marke am Blockende in drei gemeldeten Faellen (Markup-Sicht, ganz verlinkter Block, Ueberschrift), und der Seitenbezug (entsteht erst beim Setzen - weiterhin gemeldet statt geschaetzt)	M1477	M

#	Punkt	Stand	Aufw.
KDX7b	InDesign-Rest nach KDX7a: die Musterseiten stehen als Angabe im Bestand und an der Seite, ihre Elemente werden aber weder auf die Seiten uebertragen (IDML_KDOC_MASTER_NOT_APPLIED) noch zurueckgeschrieben (IDML_KDOC_MASTER_NOT_WRITTEN) — es braeuchte die Ueberschreibungs-Liste (OverrideList) und eine Abbildung auf MasterPage. Die Textrahmen-Verkettung ueberlebt als Angabe, wird aber nicht wiederhergestellt: der Text liegt seit dem Import ganz im ersten Rahmen (IDML_KDOC_STORY_CHAIN/_CHAIN_NOT_RESTORED), ihn wieder aufzuteilen hiesse zu raten, wo InDesign umgebrochen haette. Ein verankertes Objekt steht an der oberen linken Ecke seines Rahmens statt an seiner Zeile (IDML_KDOC_ANCHORED_PLACED), eine Tabelle neben Text im selben Rahmen wird uebergangen (IDML_KDOC_TABLE_IN_STORY), und Tabellen kommen ohne Zellhintergruende, Kantenlinien und Zellraender. Bezierkurven eines Umrisses werden am Ankerpunkt zum Streckenzug (IDML_CURVE_FLATTENED — das Lesemodell traegt die Richtungspunkte nicht). Nicht ausgewertet: Verlaufsfuellung, Schatten, Transparenz, Textumfluss, Objektformate, XML-Tagging und die Schriftnamen (gesetzt wird in den Standard-14). Ein nur verknuepftes Bild liegt ausserhalb des Pakets (IDML_IMAGE_LINKED)	M1477	L
ARC14	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, …)	M1478	S
ARC15	7z mehrteilig (.7z.001, .002, …) oeffnen und aufteilen — die Teile sind reine Byte-Stuecke	M1478	S
ARC16	ZIP geteilt/gespannt (.z01 … .zip) oeffnen und schreiben — Versaetze je Teil, das zentrale Verzeichnis kann ueber Teilgrenzen reichen; Gegenpruefung 7-Zip/Info-ZIP	M1478	M
ARC17	RAR mehrteilig lesen (.part1.rar, .r00) — ein Eintrag bzw. der solide Strom laeuft ueber die Teilgrenze	M1478	M
ARC18	CAB-Ketten (mehrere Kabinette) lesen und schreiben — ein Ordner-Strom setzt sich im naechsten Kabinett fort; Gegenpruefung makecab	M1478	M
ARC19	7z verschluesselt schreiben (AES-256 + SHA-256-Schluesselableitung) — das Lesen steht	M1478	S
ARC21	ISO mit El Torito (bootfaehige Abbilder) lesen und schreiben	M1478	S
ARC23	BZIP2-Packer (Sortierung aller zyklischen Verschiebungen, Huffman-Tabellenwahl) — dann auch .tar.bz2 und BZIP2 im ZIP schreiben	M1478	M
ARC26	RAR verschluesselt lesen (RAR5 AES-256/PBKDF2, RAR4 AES-128) und die Wiederherstellungsdaten	M1478	M
KMU18	Noten fuer ein anderes Instrument umschreiben (Nutzerwunsch): eine Stimme auf ein Zielinstrument uebertragen, so dass sie dort wirklich spielbar ist. Drei Bausteine liegen bereit und werden nur verkettet: Transposition (M1404, diatonisch korrekt ueber Stufenversatz und Halbtoene), InstrumentRangeCheck und PartExtractor.Extract(concertPitch). Neu zu bauen: (a) Oktavversetzung der Stellen, die ausserhalb des Zieltonumfangs liegen — samt Bericht, welche Stelle versetzt wurde, statt sie still danebenliegen zu lassen; (b) Schluessel und Notenzeilen-Zahl des Ziels uebernehmen (Instrument.StaffClefs/StaffCount — eine Bratschenstimme braucht den Altschluessel, ein Klavier zwei Zeilen); (c) Spieltechnik, die beim Ziel keinen Sinn ergibt, melden (Bogenstrich am Klavier, Pedal an der Violine); (d) Akkorde auf ein einstimmiges Instrument melden statt still zu kuerzen. Ausdruecklich NICHT: Stimmen neu verteilen, Akkorde umlegen oder eine Begleitung erfinden — das ist Bearbeitung, nicht Umrechnung, und waere geraten	M1476	M

#	Punkt	Stand	Aufw.
KMU12	Die Wiedergabeschicht: Tempo als Zahl (BPM) je Zeitstelle, Tempo- und Dynamik-Kurven, Swing, Wiederholungs-Ausfuehrung als abgeleitete Reihenfolge; ScoreToMidi ueber den vorhandenen Midi/MidiWriter — kein zweiter MIDI-Schreiber. Heute fuehrt kein Weg von Score nach MIDI	M1476	L
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	M1479	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	M1479	M
FCR3	Zeichenwerkzeug: neue Umrisse ziehen, Punkte einfuegen und loeschen, zwischen Ecke / weich / tangential umschalten, Pfad schliessen — der Punkt, ohne den es ein Betrachter bleibt	M1479	L
FCR4	Bezier-Griffe mit Stetigkeit: Kontrollpunkte ziehen, und ein weicher Punkt haelt seine Griffe kollinear — sonst entsteht bei jedem Zug ein sichtbarer Knick	M1479	M
FCR8	Glyphen-Uebersicht: alle Glyphen als Raster, nach Unicode-Bereich gefiltert, Glyphe anlegen / loeschen / umbenennen — und sichtbar, welche Zeichen noch fehlen	M1479	M
FCR10	Probe und Wasserfall: eigener Text in mehreren Groessen, vorher/nachher nebeneinander, gerastert in echten Bildschirmgroessen — der einzige Weg, eine Aenderung zu beurteilen	M1479	M
LR9	Nicht-destruktiv: die Bearbeitung ist ein Rezept, das Original bleibt unberuehrt; Sicherung als XMP-Sidecar	M1480	M
LR10	Vorgaben (Presets) + Stapel-Synchronisierung ueber viele Bilder	M1480	M
LR12	Export: Format/Groesse/Qualitaet, Ausgabeschaerfung, Wasserzeichen, Auswahl der mitgegebenen Metadaten	M1480	M
LR4	Details: Schaerfen (Betrag/Radius/Details/Maskieren) und Rauschreduzierung (Luminanz + Farbe)	M1480	L
LR5	Objektivkorrektur: Verzeichnung, chromatische Aberration, Vignettierung — rechnerisch und ueber Profile	M1480	L
KMU6	Notenkopfform, Schlagzeug, Cue-Noten: NoteheadShape (Kreuz, Raute, Schraeg, unsichtbar), Zeilenzahl je Notenzeile (1 oder 5), Percussion-Mapping, Cue-Kennzeichen an der Note; MusicXML unpitched richtig deuten	M1476	L
KMU8	Spieltechnik: Arpeggio, Glissando und Portamento, Flageolett, Bogenstrich, Daempfer, Tremolo (ein- und zweistimmig) als Auszeichnung an der Note; Stich und SMuFL	M1476	L
KMU9	Mikrotonalitaet und Cluster: Pitch.Alter als Bruch oder Cent, Accidental um Viertelton-Stufen erweitern, Cluster als eigene Form	M1476	M
LPP10	Diagramm-Ausschmueckung + SmartArt ohne mitgelieferte Zeichnung — der Rest von LPP8	M1481	L (Nische)
ROB1	JPEG 2000: Paketkoepfe, Code-Block-Zahlen, Qualitaetsschichten und Tile-Parts nehmen Zahlen aus dem Strom; der MqDecoder ist derselbe wie bei JBIG2, die Abbruchbedingung aus M1451 also unmittelbar anwendbar. Zuerst messen (Sonde je Abschneide- und Verfaelschungsfall), dann verdrahten	M1481	M

#	Punkt	Stand	Aufw.
ROB2	AV1 und HEVC: Symbol-, Kachel- und Segmentzahlen aus dem Bitstrom; eigene Leser (Av1BitReader, HEVC-Kette) mit eigener Randbehandlung — die Bedingung muss dort neu formuliert werden, MqDecoder gilt nicht	M1481	L
ROB4	Die uebrigen Bildformate: PSD-Kanaele, TIFF-Streifen und -Kacheln, GIF-/PNG-Lauflaengen, Netpbm-Masse, JPEG-Neustartintervalle — je die Stelle finden, an der eine Stromzahl eine Schleife treibt	M1481	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	M1479	L
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)	M1479	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	M1479	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	M1479	M
KDX24	FlowDocument.Image vergibt keinen SourcePath — ein allgemeiner Befund aus KDX6b, nicht auf Excel beschraenkt: jedes ueber die Fluss-Fassade gebaute Dokument mit Bild ist so nicht als .kdoc speicherbar, weil die Ressource keinen Namen hat. Fuer Excel ist es oertlich in XlsxToKdoc behoben; die allgemeine Behebung ist eine Signaturaenderung an FlowDocument.Image/ImageBlock mit Wirkung auf alle Aufrufer und gehoert deshalb in einen eigenen Durchgang	M1482	M
KDX22	PowerPoint-Restposten nach KDX21: die Platzhalter aller Folienlayouts liegen in einer Liste, und der Schreiber erzeugt ein slideLayout — definieren zwei Layouts body idx=1 an verschiedenen Stellen, bekommt eine der Folien nach dem Rundlauf die falsche geerbte Lage. Es braucht PptxPresentation.Layouts (je Layout eigene Platzhalterliste), PptxSlide.LayoutIndex, N slideLayout-Teile samt Folien-Beziehungen, sldLayoutIdLst und Content-Types, dazu das .kdoc-Feld und die Seiten-Zuordnung. Dazu die Themenfarbe je Lauf: PptxReader.ReadRun loest schemeClr sofort auf und speichert nur den Hex, PptxRun hat kein Verweisfeld — ein farbig ausgezeichneter Textlauf folgt damit keinem Themenwechsel (an der Form tut er es seit KDX21)	M1482	M
KDX23	Excel-Restposten nach KDX6b: Diagramme und Pivot-Tabellen stehen weiter nur in der Ansicht (XLSX_KDOC_CHART_VIEW_ONLY/_PIVOT_VIEW_ONLY) - bewusst nicht in den Bestand gelegt, weil der Schreiber sie gar nicht ausgeben koennte: XlsxWriteChart hat keinen Quellbezug (c:f, das Diagramm kaeme ohne Bindung an die Zellen zurueck), und XlsxWriter kennt keine Pivot-Teile (pivotCacheDefinition, pivotCacheRecords, pivotTable, drei Beziehungen, Content-Types) - sie zu speichern waere genau das Feld, das niemand ausliest. Dazu liest schon XlsxReader Spaltenbreiten, Zeilenhoehen, Blattschutz, Kopf-/Fusszeilen und Makros nicht; Ansicht -> Bestand bleibt einseitig (eine Aenderung an den gesetzten Tabellen wirkt nicht auf die Mappe - dafuer muesste jede Zelle als eigenes Objekt mit ihrem Zellbezug im Layout liegen); und eine fehlende Bild-Ressource des Tabellen-Bestands wird erst beim Lesen gemeldet statt schon beim Schreiben wie bei Bildrahmen	M1482	L

#	Punkt	Stand	Aufw.
KDX25	Der Seitenbezug in der Export-Sicht (der eine Rest von KDX19): die Export-Sicht ist eine flache Blockliste — auf welcher Seite ein Block landet, entsteht erst beim Setzen. Ein Kommentar, eine Note oder ein Abschnittswechsel wird deshalb weiterhin gemeldet statt geschaetzt; eine falsche Seitenzahl waere schlimmer als keine. Es braeuchte einen Rueckkanal vom Satz in die Export-Sicht (wie beim Inhaltsverzeichnis der DevDoc, M111: Seitenzahlen im Fixpunkt zurueckgelesen). Klartext haengt weiterhin bewusst nichts an — eine blosse Markerziffer ohne ihren Text saehe wie Teil des Satzes aus, und reiner Text hat keinen Seitenfuss	M1482	M
KMU20	Bedienbar machen, was der Stich laengst setzt (Restposten nach KMU15): Gabeln (Crescendo/Decrescendo), Verzierungen, Oktavierung, Pedal, Atemzeichen, Tempo- und Vortragsangaben, Vorschlagsnoten, N-tolen, Haltebogen, Balkensteuerung, erzwungene Halsrichtung, Akkordsymbol, Fingersatz, Generalbass und die mehrtaktige Pause — alle im Modell und im Notenstich vorhanden, im Editor nicht bedienbar. Dazu: die Pfeiltasten verschieben nur den Grundton statt des ganzen Akkords, und der Import laesst ab der dritten Stimme und der vierten Notenzeile fallen (gemeldet)	M1476	M
KMU19	Die Eingabezeile zeigt, was der Satz zeigt (Restposten nach KMU15): sie zeigt weiter eine Notenzeile, eine Stimme und vom Akkord nur den untersten Kopf — gewechselt wird ueber Knoepfe. Ein Notenbild mit beiden Zeilen uebereinander und beiden Stimmen nebeneinander ist der naechste Schritt, und dabei muss die Trefferermittlung wissen, in welche Zeile geklickt wurde (Zeichnen und Treffen nehmen dieselbe Geometrie — die Regel aus M1421/M1351). Dazu: die Zahl der Notenzeilen kommt aus dem Instrument und laesst sich nicht dagegen einstellen (Klavier einzeilig geht nicht), zwei Stimmen sind die Obergrenze (ab drei setzt der Stich die Halsrichtung nach der Lage), je Note ein Artikulationszeichen und eine Silbe der ersten Strophe. Gestaltung erst nach dem Profi-Vorbild, das der Nutzer waehlt	M1476	L
MUS14	Anhoeren: die Partitur als Klang wiedergeben (MIDI-Ausgabe an das System bzw. ein einfacher eigener Synthesizer fuer die Vorschau)	M1483	L
ARC25	UDF schreiben — Gegenpruefung Linux-Kern und mkudffs	M1484	M
ARC34	BZIP2-Packer verbessern (Restposten nach ARC23): es werden zwei gleiche Huffman-Tabellen geschrieben (die Norm verlangt mindestens zwei). Das echte bzip2 waehlt bis zu sechs und teilt den Block in Abschnitte — bei gemischten Daten packt es dadurch besser. Die Ausgabe ist gueltig (bzip2 1.0.8 nimmt jede erzeugte Datei an, bei Text ist sie sogar kleiner), nur nicht so dicht	M1484	S
ARC35	Restposten an Verbund und Bootabbild: beim ZIP-Teilen wird nur zwischen Eintraegen geschnitten, und ein Verbund ueber 4 GB braeuchte ZIP64-Versaetze beim Vereinigen (ARC16); beim El-Torito-Schreiben entsteht ein Booteintrag, mehrere Abschnitte BIOS+EFI nebeneinander gehen nur beim Lesen (ARC21)	M1484	S
ROB3	Archive: Eintragszahlen, unkomprimierte Groessen und Lauflaengen in ZIP/7z/RAR/TAR/CAB/ISO. Hier ist die Eingabe-zu-Arbeit-Spreizung strukturell am groessten (Dekompressionsbombe), und ein Deckel je Format greift zu kurz — gebraucht wird ein Verhaeltnis aus verbrauchten Eingabebytes zu erzeugten Ausgabebytes	M1484	L
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	M1485	XL
FCR12	Raster, Fang und Messen: an Hilfslinien, ganzen Einheiten und dem Em-Raster fangen; Abstaende messen und bemassen	M1485	M

#	Punkt	Stand	Aufw.
FCR11	Rueckgaengig-Kette und Zwischenablage: je Glyphe und dokumentweit; Glyphen zwischen zwei Schriften kopieren	M1485	M
FCR15	Zeichnung uebernehmen: einen SVG-Pfad oder ein Vektorzeichen aus einem PDF als Glyphe einsetzen — der uebliche Weg, wie ein Logo zur Schrift wird	M1485	M
FCR17	Restposten in der Glyphen-Uebersicht (nach FCR8): eine Glyphe anlegen und loeschen fehlt, und beides aus einem Grund, nicht aus Bequemlichkeit — fuers Anlegen gibt es keinen Kern (ein neuer Glyphenplatz muss maxp/loca/hmtx mitwachsen lassen), und Loeschen ginge nur ueber FontGlyphSetEditor.ExtractSubset, das alle Glyph-IDs umnummeriert und GPOS/GSUB verwirft — das heimlich einzubauen haette eine Schrift beschaedigt, die Ligaturen oder Kerning traegt	M1485	M
LR11	Katalog: Ordner/Sammlungen, Bewertung/Markierungen/Stichwoerter, Suche und Filter	M1486	L
LR13	Kontaktbogen / Web-Galerie / Fotobuch	M1486	M
LR16	Restposten des Exports (nach LR12): Text-Wasserzeichen fehlen — sie braeuchten einen Glyphen-Rasterer, und Photo darf weder Fonts noch Rendering importieren (die Architektur-Grenze ist der Grund, nicht die Muehe); die Ausgabeschaerfung unterscheidet kein Medium (Bildschirm/mattes/glaenzendes Papier) und rechnet nicht gegen die Zielaufloesung; Metadaten werden nur in JPEG und PNG mitgegeben (andere Behaelter werden benannt uebergangen), IPTC gar nicht, solange es nicht feldweise filterbar ist	M1486	M
LR17	ImageConverter.ToRgba liest nur PNG, JPEG und TIFF (sonst throw) — ein allgemeiner Nebenbefund aus LR12, aelter als diese Arbeit: er betrifft convert_image genauso, und die Foto-Werkzeuge erben die Grenze. Ueber RasterImage.Parse kommen dagegen alle Bildformate herein; die zwei Wege sind auseinandergelaufen	M1486	S
IMG23 8/8	JBIG2 schreiben — gescannte Schwarz-Weiss-Seiten deutlich kleiner als CCITT G4	Untertabelle unten: IMG23.1–23.8	L

#	Punkt	Stand
IMG7.9	Transform-Skip	M1487
IMG7.7	Schleifenfilter beim Kodieren (Deblocking + SAO)	M1408 - Deblocking laeuft per Vorgabe (pps_deblocking_filter_disabled_flag = false), der SAO-Entscheidungskern ist bis in den Bitstrom verdrahtet (slice_sao_luma/chroma_flag, zweiter Kodierdurchgang) und gegen libheif belegt (HeifSaoLibheifOrakelTests, hoechstens 1 von 255 daneben). Die Zeile stand bis M1489 auf 'Deblocking offen, Filter im Slice-Header abgeschaltet'
IMG7.8	Volle Modus- und Quadbaum-Entscheidung (Rate-Distortion)	M1491 - Schalter rateDistortion (Vorgabe aus): Kodier-Quadbaum, Modus (Hadamard-Vorauswahl, dann voll gerechnet) und Transformbaum-Tiefe nach D + ·R, Bits aus einem Zaehlmodus des CABAC-Codierers. Gegen ffmpeg und libde265 Abweichung 0; bei jeder Stufe und jedem Motiv kleiner UND besser (Viertelmuster: 2012 -> 908 Bytes, 35,7 -> 40,9 dB), zum Preis der rund zehnfachen Laufzeit
IMG7.10a	4:4:4 und 4:2:2 schreiben	M1491 - HevcChromaFormat, RExt-Profil; gegen ffmpeg und libde265 Abweichung 0 auf den rohen Ebenen. Die Orakel fanden drei eigene Fehler: die 4:2:2-Modustabelle (276 Abtastwerte daneben), das Kantenglaetten des eigenen Lesers und eine Ausnahme bei zwei Winkelmodi
IMG23.5	Verfeinerungs-Region schreiben	M1487

#	Punkt	Stand
IMG23.6	Halbton-Region schreiben	M1487
IMG29.7	Sukzessive Annaeherung beim progressiven Schreiben	M1487

#	Punkt	Stand	Aufw.
ARC24	LZX-Packer fuer CAB	M1489	M
ARC32	archive_extract bei soliden RAR (der Rest von ARC20 — fuer 7z ist er erledigt): ein solide gepacktes RAR wird heute benannt abgelehnt, es ist gar nicht lesbar; da ist also nichts an Laufzeit zu sparen, wie die Zeile annahm, sondern es fehlt das sequenzielle Entpacken mit ueber die Eintraege durchgereichtem Woerterbuch	M1489	M
ARC33a	RAR mit verschluesselten Koepfen (-hp) — dort ist schon die Dateiliste unlesbar — und „verschluesselt UND mehrteilig“. Der Aufbau ist gemessen, nicht vermutet: ein unverschluesselter Typ-4-Kopf, danach jeder Satz mit eigenem 16-Byte-Startwert davor, CBC, auf 16 aufgefuellt — kein durchgehender Strom; die Nutzdaten dazwischen tragen ihre eigene Verschluesselung. Der Entwurf ist ein entschluesselter Abzug, damit der gewoehnliche Leser keine Sonderbehandlung braucht. Orakel WinRAR 7.22 (-hp, -v -p) — die Zeile hatte „ohne Orakel nicht belegbar“ behauptet, und das liess sich erzeugen	M1489	M
ARC36a	Mehrere El-Torito-Abschnitte schreiben (BIOS+EFI nebeneinander) — gelesen wurden sie schon. Der Entwurfspunkt sind zwei Rollen statt einer Liste: El Torito kennt genau einen Vorgabeeintrag, der seine Plattform aus der Gueltigkeitszeile nimmt; jeder weitere Abschnitt bekommt einen eigenen Kopf (0x90/0x91) mit eigener Plattform. Dabei fiel auf, dass der Leser gar keine Plattform je Eintrag nannte — die MCP-Auflistung druckte fuer jeden Eintrag die der Gueltigkeitszeile, bei BIOS+UEFI also zweimal „X86“. Orakel xorriso 1.5.4 in beide Richtungen, dazu haengt der Linux-Kern das eigene Abbild ein	M1489	S
KDX27	Pivot-Tabellen im Excel-Bestand (der Rest von KDX23 - Diagramme samt Quellbezug sind seit M1482 drin): der XlsxWriter kennt keine Pivot-Teile (pivotCacheDefinition, pivotCacheRecords, pivotTable samt drei Beziehungen und Content-Types). Sie in den Bestand zu legen, bevor der Schreiber sie ausgeben kann, waere genau das Feld, das niemand ausliest - erst der Schreiber, dann der Bestand; bis dahin bleibt XLSX_KDOC_PIVOT_VIEW_ONLY die ehrliche Meldung. Dazu die kleineren Reste: je Diagramm eine Datenreihe (gemeldet), vom Blattschutz nur das Kennzeichen, keine erste/gerade Kopfzeile, und Ansicht -> Bestand bleibt einseitig	M1489	M
KMU7	Tabulatur und Griffbilder: eine eigene Notenzeilen-Art fuer Tabulatur mit Saiten- und Bundangabe, Akkordraster als eigener Typ; Stich dafuer	M1489	L

#	Punkt	Stand	Aufw.
KMU16	Restposten nach KMU1: am MusicFrame ueberleben Rules und MusicFont den .kdoc-Rundlauf nicht (nur Score und SuppressMusicFont) — die vollen EngravingRules (~40 Angaben) und die Systemgruppen gehoeren zu KMU10/KMU14, MusicScoreInfo traegt heute die fuenf Angaben, die der Editor wirklich verliert. Ein Rahmen zeigt eine Satzseite; braucht die Partitur mehr, wird der Rest nicht gezeichnet und ueber MUSIK_UMBRUCH_FEHLT gemeldet. Dazu fehlen MCP-Werkzeug und Pruefstand-Beispiel fuer den Noten-Datenteil — bewusst erst nach KMU2 nachzuziehen, weil die Weichen dorthin gehoeren (die Fehlerklasse IMG1/IMG22/IMG27: gebaut, aber nicht ansteuerbar)	M1489	M
KMU17	Mehrseitige Ansicht (Restposten nach KMU2): ein MusicFrame zeigt den ersten Satzblock, die Ansicht ist EINE Seite mit EINEM Rahmen — je Satzblock eine Layout-Seite haette heute geheissen, die ganze Partitur je Seite erneut ins JSON zu schreiben (der Rahmen haelt sie inline). Der Bestand ist davon nicht betroffen, die Kuerzung wird ueber EngravingCodes.NoLineBreaking gemeldet. Es braucht einen Seitenindex am Rahmen und einen gemeinsamen Bestandsverweis statt der Inline-Kopie. Dazu: .mxl ist kein Exportziel (der Schreiber schreibt nur unkomprimiert, es fehlt der ZIP-Behaelter samt META-INF/container.xml), und der MusicXmlReport fuehrt seine Befunde als blossen Klartext ohne Schluessel — die Fundgrube nutzt deshalb das Sammelmerkmal MUSICXML_LESEVERLUST	M1489	M
KMU21a	N-tolen: der Rechenfehler an der Pause und die N-tole ueber den Taktstrich. Die Zeile verschwieg den Kern: Tuplets stand nur an der Note, eine Pause in einer Triole behielt ihre geschriebene Dauer — der Takt war um den N-tolen-Faktor zu lang, still und zugleich in Voice.Quarters, der Spaltenrechnung, der Wiedergabe und der MusicXML-Ausgabe. Tuplets sitzt jetzt am MusicElement (eine Stelle); music21 bestaetigt quarterLength 2/3 statt 1, den Takt mit genau 4.0 statt 4.333. Dazu die Gruppierung je Notenzeile statt je Takt (im XML-Leser lebt der Verfolger an der Stimme; music21 liest die Gruppe ueber den Taktstrich als EINE), die Klammer folgt der Balkenneigung (derselbe Deckel wie beim Balken, die Grundhoehe so gesucht, dass jedes Glied freikommt), und die Ziffernbreite kommt aus der Schrift statt aus einer Naeherung	M1489	M
KMU22a	Volte ueber den Zeilenumbruch: in der Folgezeile mit der Ziffer in Klammern und ohne Anfangshaken, wie der klassische Stich es haelt. Die alte Zeile war hier ungenau: die Volte wurde doch fortgesetzt — nur ohne Ziffer, und nur, wenn der Takt dort selbst eine Volten-Angabe trug (die steht in der Praxis nur am Anfang und am Ende)	M1489	S

#	Punkt	Stand	Aufw.
KMU23	Restposten nach KMU11: die Fussnote geht nicht durch MusicXML — das Format kennt keinen Begriff dafuer, und sie in ein words zu zwingen hiesse, sie als etwas auszugeben, was sie nicht ist; sie ueberlebt ueber das .kdoc. Dazu ein Befund aus dem Bauen: MusicXmlWriter.Write hat gar keinen Bericht (er gibt nur Bytes zurueck) — der Verlust steht deshalb nur in der Klassendoku statt in einer Laufzeitmeldung, die einzige Stelle im Musikteil, an der die Regel „nie stillschweigend" nicht eingehalten ist. Die Widmung ueberlebt unseren Rundlauf, aber nicht den ueber music21 (das uebergeht einen credit-Block beim Lesen vollstaendig — gemessen, Grenze des Fremdwerkzeugs). Beischriften weichen einander nicht aus und verbreitern die Spalte nicht (Fussnotenmarke, mehrere Spielanweisungen an derselben Zeitstelle, Dynamik und Gabel in derselben Zeile); der Titel wird nicht zentriert, der Rechtevermerk steht nur auf Seite 1, laufende Kolumnentitel gibt es nicht	M1489	M
KMU28	Restposten nach KMU8 (Spieltechnik): das Glissando wird gerade statt gewellt gezogen (MUSIK_GLISSANDO_GERADE) — SMuFL fuehrt keine Wellenlinie beliebiger Laenge und Neigung, nur gestapelte Zeichen ohne Neigung, und eine schiefe Kette saehe nach Fehler aus. Die Tremolo-Striche stehen waagerecht, weil die Balkenneigung erst nach dem Balkensatz feststeht (dieselbe Grenze wie die N-tolen-Klammer, KMU21)	M1489	S
MUS12a	Notensatz als Bild und Metafile: ScoreExport setzt die Partitur als LayoutDocument (je Satzblock eine Seite) und reicht sie an DocumentExporter — kein zweiter Zeichenweg; PDF, PNG, TIFF und EMF gehen, und belegt ist, dass dieser Weg dieselben Notenlinien liefert wie MusicPdf.Render. Dazu MusicFrame.PageIndex/EngravedPageCount/UseDocumentScore fuer die mehrseitige Ansicht	M1489	M
ARC20	archive_extract bei soliden 7z/RAR einmal auspacken statt je Eintrag neu	M1488 - ArchiveReader.ExtractAll packt ein solides 7z/RAR in EINEM Vorgang aus, archive_extract ruft es (bei der Pruefung M1489 gefunden: die Zeile stand noch auf offen)	S
BSP1	Photo ist im Pruefstand unsichtbar: zehn MCP-Werkzeuge (photo_develop/_batch/_preset/_recipe/_sheet/_catalog/_tag/_collection/_book/_gallery), aber keine Seite und ein Beispiel nur fuer photo_sheet — obwohl seit M1480/M1486 Entwicklung, Vorgaben, Katalog, Galerie, Fotobuch und Objektivkorrektur stehen	M1490 - Pruefstand-Seite /foto: Bild hochladen, alle Regler des Entwicklungsrezepts, Ausgabeschaerfung und Rauschreduzierung, Vorgabe als XMP laden, Vorschau, Download als PNG/JPEG/TIFF und Rezept als XMP-Sidecar; Rechenkern Model/FotoStudio ruft dieselben Core-Klassen in derselben Reihenfolge wie photo_develop. Im Browser nachgemessen (Belichtung +1,5 hebt die mittlere Helligkeit 101 -> 160). Bewusst nicht auf der Seite: Objektivkorrektur, Geometrie, Wasserzeichen und Metadaten-Export - sie gehoeren nicht zum Rezept und bleiben ueber MCP erreichbar	M

#	Punkt	Stand	Aufw.
BSP2	MusicOcr hat kein MCP-Werkzeug und kein Beispiel; erreichbar ist die Notenblatt-Erkennung nur ueber /noteneditor. Genau die Fehlerklasse IMG1/IMG22/IMG27: gebaut, aber nicht ansteuerbar	M1490 - MCP-Werkzeug music_ocr (PDF oder Bild -> MusicXML und/oder .kdoc, dazu der Erkennungsbericht; bei 'nichts erkannt' wird nichts geschrieben, der Bericht sagt es) an allen Drift-Stellen registriert, durch den echten Handler getestet; DevDoc-Beispiel NotenerkennungBibliothek. Nur drei Optionen nach aussen (clef, ignore_coloured_ink, font)	S
BSP3	Duenne Bereiche, gemessen am Codeumfang: MusicXML (1) und MIDI (1) zeigen die Musik-Faehigkeiten nicht als Austauschformat; danach Archive (2), Barcodes (2), Rechnung (3) und OCR (4) — gezaehlt M1489 an ExampleCatalog (360 Eintraege). Der Notensatz ist mit 9 Beispielen kein duenner Bereich mehr	M1491 - MusicXML 1 -> 4, MIDI 1 -> 4, Archive 2 -> 5, Barcodes 2 -> 5 Beispiele, alle ueber die vorhandene API	M
BSP4	Ein Waechter gegen Duennheit fehlt. Der vorhandene verlangt, dass jedes registrierte Beispiel sichtbar ist — nicht, dass ein Namensraum mit dreissig oeffentlichen Typen mehr als ein Beispiel hat. Genau daran ist der Notensatz siebzig Meilensteine lang bei einem Beispiel geblieben, ohne dass etwas rot wurde. Ein Verhaeltnis aus oeffentlichen Typen je Bereich zu Beispielen waere die Regel — mit begruendeter Ausnahmeliste, wie beim DevDoc-Waechter (M111)	M1491 - DevDocExampleDensityGuardTests: mindestens ein Beispiel je angefangene 20 oeffentliche Typen, mindestens zwei je Bereich, Ausnahmeliste leer. Vor BSP3 rot an genau den vier duennen Bereichen und an keinem der uebrigen 18 - die Schwelle wurde nicht an den Bestand angepasst	M
LR7a	Lokale Anpassungen mit Masken: Verlaufs- und Radialfilter, Pinsel, verknuepfbar	M1491 - PhotoMask/LinearGradientMask/RadialGradientMask/BrushMask, LocalAdjustment ueber dieselben Reglerfunktionen wie global, gemischt im linearen Licht, im Rezept nach der globalen Entwicklung; Lightroom-XMP (crs:GradientBasedCorrections/CircularGradientBasedCorrections) in drei Schreibweisen. Orakel numpy und exiftool 12.76 (fand klein geschriebene Wahrheitswerte). Bedienbar auf /foto (im Browser: Radialfilter +2 hebt die Mitte 112 -> 211, Ecke unveraendert) und ueber photo_develop per Sidecar; ohne Masken byte-identisch	XL

#	Punkt	Stand	Aufw.
LR7b	Restposten der Masken: Bereichsreparatur, Pinsel im Sidecar, Midpoint/Roundness, Weiss/Schwarz/Dunst/Tonung lokal, DevelopSync mit lokalen Anpassungen	M1495 - Bereichsreparatur SpotRemoval (Kreis oder gemalter Bereich, Kopieren/Heilen, Weichheit, Deckkraft, im Rezept vor jeder Tonwertrechnung) mit Heilen als eigenem Baustein PoissonHealing (Gradientenangleich nach Pérez/Gangnet/Blake, Membran per konjugierter Gradienten im linearen Licht; gegen scipy spsolve derselben Diskretisierung auf das Byte) und automatischer Quelle (WithAutomaticSource, aehnlichster Umgebungsring unter festen Kandidaten). Sidecar: crs:RetouchAreas (Ellipse/Pinsel) lesen+schreiben, crs:RetouchInfo lesen - die Feldzuordnung ist von Adobe selbst belegt (Camera Raw 9.1 schreibt dieselben Kreise in beiden Formen: SizeX=SizeY=radius, SourceX/OffsetY = Lage der Quelle). Pinsel crs:PaintBasedCorrections/Mask/Paint/crs:Dabs lesen+schreiben; der Strichradius bezieht sich jetzt auf die Bildbreite (Belege: zwei echte Adobe-Ausgaben aus dem Adobe-Forum, Aussage+Messung eines Vorgaben-Autors auf vmoldo.com, Adobes Doppeleintrag); viele Tupfer laufen ueber ein Gitter, genau gleich wie Strich fuer Strich. Midpoint/Roundness (Richtung nach Adobes Vignetten-Hilfe), BasicAdjustments.Toning + lokale Weiss/Schwarz/Dunst/Tonung (crs:LocalWhites2012 usw.), DevelopSettings.LocalAdjustments/SpotRemoval/Global. Bedienbar auf /foto (im Browser: Fleck 230 -> 100 mit Reparatur, ohne 230) und ueber photo_develop (spots), photo_recipe, photo_batch (local, spots). Festlegungen, gegen Lightroom nicht pruefbar (kein Lightroom hier): M beginnt einen Strich, f/h setzen Fluss/Haerte nach dem Muster von r, Weichheit = 100·(1CenterWeight), die Formeln von Mittelpunkt/Rundheit/Tonung, das Heilverfahren (Adobes Method="gaussian" ist nicht veroeffentlicht), die Quellsuche; ob der Radius bei Hochformat auf Breite oder lange Kante zielt, trennt kein gefundenes Beispiel (alle quer). Benannt abgelehnt: ovale/gedrehte/abgestufte Reparaturellipsen, mehrere Masken je Reparatur, andere Reparaturarten. Masken-Gruppen ab Lightroom 11 -> neue Zeile LR7c	L

#	Punkt	Stand	Aufw.
LR7c	Masken-Gruppen ab Lightroom 11 (crs:MaskGroupBasedCorrections) samt Mischformen, Bereichs- und KI-Masken	M1499 - Lesen und Schreiben ueber das vorhandene Modell (LocalAdjustment/PhotoMask, kein zweiter Weg); geschrieben wird jede Anpassung in die Masken-Gruppen (eine Liste, Reihenfolge erhalten), gelesen auch die drei alten Listen. Verknuepfung nach Messung: Camera Raw (photoshop-mcp, MIT) und echte Lightroom-Sidecars - Abziehen ist MaskBlendMode 1 + MaskValue 0, Schneiden „die Umkehrung abziehen", Modus 2 ungueltig (benannt abgelehnt), MaskValue 0 im Pinsel der Radierer; MaskInverted statt Flipped. Neu: MaskGroup (Mask/Aggregate), BidirectionalGradientMask (Camera Raw 18.6), LuminanceRangeMask (LumRange-Trapez, Reihenfolge nach Lightroom-Classic-15-Messung, photo-workflow-mcp), ColorRangeMask (PointModels/AreaModels), DepthRangeMask (braucht die Tiefenkarte), ImageMask (KI-Maske; Verweis und alle Lightroom-Angaben bleiben, Anwendung ueber ein gesetztes MaskRaster, ohne Raster benannt abgelehnt), die aeltere Bereichsangabe an der Korrektur (Lightroom 7.3-10) als Schnitt, MaskActive/MaskName/CorrectionName. Zwei Skalenfehler behoben (an Camera Raw 18.6 gemessen): LocalExposure2012 ist geviertelt (vorher ein Viertel der Wirkung), CorrectionAmount reicht bis 2 und skaliert die Regler. Bedienbar auf /foto (Luminanzbereich anlegen/begrenzen, KI-Raster hochladen, Maskenbeschreibung, Maskenansicht mit dem Bild - im Browser belegt) und ueber photo_develop (local, mask_rasters, depth_map) und photo_recipe (Masken je Anpassung). Festlegungen, gegen Lightroom nicht pruefbar: Luminanz als CIE L*/100 (gestuetzt: Grenze 0,40 lag auf einem Graukeil bei Tonwert 91, L* 38,7), Uebergaenge per Smoothstep, Farbbereich mit Bezugsfarbe aus dem Bild an der Messstelle und E*76-Toleranz 5+45·Verfeinern (Lightrooms drei Farbwerte sind kein sRGB und werden nur zurueckgeschrieben), Lum/DepthMin/Max/Feather als Trapez mit Rampe Feather/2, die Mitte des beidseitigen Verlaufs, MaskValue zwischen 0 und 1 im Modus 0 als Deckkraft, SyncIDs aus der Lage. Benannt offen -> LR7d (.acr-Raster, nicht belegbar) und LR7e (uebrige lokale Regler)	M

#	Punkt	Stand	Aufw.
LR7e	Lightrooms uebrige lokale Regler (LocalTexture, LocalHue, LocalGrain, LocalLuminanceNoise, LocalMoire, LocalDefringe, MainCurve, LocalCurveRefineSaturation, LocalPointColors/LocalColorVariance, LocalColorGrade*)	M1500 - lesen, schreiben und anwenden ueber die vorhandenen lokalen Anpassungen und Masken (kein zweiter Weg; voll deckend byte-gleich mit dem globalen Baustein). Neue globale Bausteine: BasicAdjustments.Texture (Mittelton-Kontrast auf log-Luminanz, fester Radius, gestauchtes Detail - Verhalten nach RAWmakases Camera-Raw-Messung), BasicAdjustments.HueShift, NoiseReduction.Moire (Farbdifferenzen ohne Kantenschutz geglaettet, derselbe Weg wie Farbrauschen), LensCorrection.Defringe (Purpur/Gruen an harten Kanten), ToneCurve.ApplyRefined (Saettigung verfeinern genau nach RAWmakases Camera-Raw-Messung), ColorGrade (Tiefen/Mitteltoene/Lichter/Gesamt, Ueberblendung, Abgleich), PointColor/PointColorRange (19-Zahlen-Format und Wirkung nach RAWmakase, an der Camera-Raw-Ausgabe gegengeprueft: Saettigungs- und Luminanzpunkte liegen genau 0,18/0,73 um die gewaehlte Farbe); Rauschen und Koernung ueber die vorhandenen NoiseReduction.Luminance/PhotoEffects.ApplyGrain. Einheiten belegt an einer Camera-Raw-18.6-Ausgabe samt Messung am Regler (photoshop-mcp, MIT): /100, LocalHue /180, LocalColorGrade* und LocalCurveRefineSaturation unverkleinert mit Vorzeichen, Kurve als "x,y" auf 0..255, leere Punktfarben-Wahl (19 mal -1) uebersprungen. Bedienbar ueber photo_develop/photo_batch (local), photo_recipe (Regler je Anpassung) und /foto (alle Felder, im Browser belegt: Werte aus einem Sidecar angezeigt, Struktur -100 veraendert die Vorschau); Katalog-Beispiel LokaleReglerRestBibliothek. Festlegungen, gegen Lightroom nicht pruefbar: die Formeln von Struktur, Farbton (HSL-Kreis), Moire, Farbsaum und Farbkorrektur, der Koernungs-Massstab (100 = Staerke 0,2), die Rampen und das Fenster-Wachstum der Punktfarbe, dass die Staerke Kurve/Punktfarbe/Farbkorrektur nicht skaliert; negatives Rauschen/Moire/Farbsaum nimmt in Lightroom eine globale Minderung zurueck, die das Rezept nicht fuehrt - darum ohne Wirkung. Benannt offen -> LR7f (LocalGlow, Wirkung nicht belegt)	M
KDX26	InDesign-Restposten nach KDX7b	M1492 - alle acht Teile: Zellstile, Verlauf, Deckkraft, Textumfluss, Schriftnamen, verknuepftes Bild (M1489), dazu in M1492 der Schlagschatten in beide Richtungen (DropShadowSetting LayoutObject.DropShadow; Streuung, Rauschen, Mischmodus und Teil-Schatten werden gemeldet; geschrieben mit InDesigns eigener Attributmenge aus Preferences.xml), XML-Tagging (Tags.xml/BackingStory.xml lesen, im .kdoc als xmlTags/xmlElements, zurueckschreiben - SimpleIDMLs export_xml() liefert auf unseren Paketen Zeile fuer Zeile dieselben Baeume wie auf den drei Adobe-Dateien), das verankerte Objekt an seiner Zeile und die Tabelle neben Text (beide als AnchoredObject im Textlauf, derselbe Weg wie die Formel). Ehrlich: eine benutzerdefinierte Verankerung wird an der Ankerzeile gesetzt statt versetzt und gemeldet; OverriddenPageItemProps bleibt ausgeblendet und gemeldet, weil keine Pruefdatei einen gefuellten Override traegt; fuer Schatten, Anker und Tabelle traegt keine der 18 Adobe-Dateien ein Beispiel - dort belegt der Rundlauf Selbstkonsistenz	L

#	Punkt	Stand
IMG33 M1493	Jpeg2000Encoder und der Wavelet-Leser kommen mit sehr kleinen Bildern klar (1x1, 2x1, 1xN, Nx1, 3x3 bis ueber die Code-Block-Groesse). Es waren zwei Ursachen, nicht eine: (1) CountBlocks zaehlte fuer ein leeres Teilband (Breite oder Hoehe 0) mindestens einen Code-Block — daher die ArgumentOutOfRangeException aus Jpeg2000Ebcot.Block; ein leeres Teilband hat nach der Norm keine Code-Bloecke. (2) Die mehrstufige Zerlegung brach ab, sobald eine Kante 1 wurde (cw > 1 && ch > 1), waehrend die Teilband-Geometrie alle Stufen annahm; die Norm zerlegt jede Richtung fuer sich weiter (eine Laenge-1-Zeile bleibt unveraendert). Schreiber und eigener Leser teilten diesen Fehler, die Rundlauf-Tests blieben gruen — erst OpenJPEG (Pillow) dekodierte 63x1, 2x1 und 1x5 falsch; mit der Sabotage-Probe (Abbruchbedingung zurueck) werden genau die drei Orakel-Faelle rot. Belegt: 16 Groessen x 4 Zerlegungstiefen verlustfrei bildpunktgleich, Schichten/Precincts/Kacheln/MCT+Alpha/Zielgroesse an kleinen Bildern, OpenJPEG bildpunktgleich	S

#	Punkt	Stand	Aufw.
FCR20	Ueberlappungen entfernen (der eine Rest von FCR5): eigener Rechenkern Fonts/GlyphOverlapRemover auf den Bezierkurven selbst — Teilen an den x/y-Extremstellen (monotone Stuecke, auch die kubische Schleife wird so zum Schnitt zweier Stuecke), Schnittpunkte quadratischer und kubischer Kurven per Halbierung bis zur Flachheit plus Newton, Endpunkte auf fremden Kurven teilen diese (T-Stoss, gemeinsame Kante, deckungsgleiche Teile), Knoten einrasten, deckungsgleiche Kanten zu einer fassen, Non-Zero je Kante beiderseits der Mitte (Seitenabstand hoechstens ein Viertel bis zur naechsten fremden Kante), Verketten mit erstem Ausgang im Uhrzeigersinn, Teilstuecke derselben Kurve wieder verschmolzen (der Grad bleibt: TrueType bleibt quadratisch). Laufrichtung TrueType im, CFF gegen den Uhrzeigersinn; ganze Schrift mit RemoveOverlaps(byte[]), zusammengesetzte Glyphen nur bei ueberlappenden Teilen zerlegt. Fremd-Orakel skia-pathops + fontTools: 18 Formen (Flaeche + 32x32-Raster bitgleich), ganz Noto Sans Bold (OFL, fester Schnitt wght 700): alle 2647 einfachen Glyphen, 596 davon ueberlappend, in Flaeche und Raster gleich; Bravura (CFF): 3669 Glyphen in der Flaeche gleich, 3 Raster je in einem Punkt anders, der 9,3e-6 Einheiten neben der Kurve liegt — mit Bruechen exakt nachgerechnet hat die eigene Antwort recht, skia/fontTools rechnen dort in float. Bedienbar im Schrift-Erzeuger (je Glyphe und ganze Schrift) und ueber font_edit op: remove_overlaps (mit instance fuer variable Schriften). Benannte Grenzen: eine variable Schrift wird abgelehnt — die gvar-/CFF2-Abweichungen haengen an den Punktnummern, und jeder Zwischenschnitt schneidet seine Kurven an anderen Stellen, ein fuer alle Schnitte gueltiger ueberlappungsfreier Umriss existiert im Allgemeinen nicht (Weg wie bei fontTools: erst instanziieren); geaenderte Glyphen verlieren ihre Hinting-Anweisungen (dieselbe Grenze wie jedes SetOutlines)	M1497	L

#	Punkt	Stand	Aufw.
FCR21	Restposten in Uebernahme und Glyphenplatz	M1507 - SVG: Rechteck (auch abgerundet), Kreis, Ellipse, Vieleck, Linienzug und Text (ueber die Standardschrift-Umrisse, Helvetica bzw. Courier fuer monospace) neben Pfaden; transform an Element und Gruppen; fill-rule="evenodd" (auch geerbt und in style) als Umlaufrichtung nach Verschachtelungstiefe; defs/display:none/fill:none/line liefern keine Form (Svg/SvgGlyphFormen). PDF: jede Seite waehlbar, umrandete Pfade ueber den neuen Konturierer Fonts/GlyphStroker als Flaeche ihrer Linie (Breite mit dem Massstab der Matrix, Ende und Ecke aus dem Grafikzustand). Glyphenplatz: GlyphSlotEditor.AppendGlyph(font, vorschub, name, codepoint) setzt Name (post bzw. CFF-Zeichensatz) und Zeichen (cmap, Format 4 oder 12) in einem Schritt, und das Anlegen geht jetzt auch in CFF-Schriften; dabei behoben: der CFF-Neubau (auch hinter SetOutlines) verlor die Glyphennamen. Bedienbar im Schrift-Erzeuger (Seitenwahl, Name/Zeichen beim Anlegen, im Browser belegt) und ueber MCP font_edit op: add_glyph. Ehrlich: evenodd ist exakt fuer Konturen, die einander nicht schneiden; use, text-anchor und tspan-Lagen werden nicht ausgewertet; Strichelungen werden durchgezogen; Kurven werden fuer den Linienrand in Strecken zerlegt; eine Flaeche mit Kontur (B) wird nur als Flaeche genommen; CFF-Neubau ohne Hinting/Subroutinen (wie jede CFF-Umriss-Aenderung)	M
FCR19	Die Schrift-Faehigkeiten ueber REST und die benannten Grenzen der Schrift-Editoren	M1508 - 19.1 REST /font-info, /font-edit, /font-convert (OpenAPI; Waechter: jeder MapPost-Pfad steht im Dokument); die Uebersetzung einer Bearbeitungs-Anfrage liegt jetzt im Kern (Fonts/FontEditJson) und MCP font_edit wie REST nutzen sie. 19.2 SetComposite: Ring ueber jede Stufe erkannt, maxp (Teile, Tiefe, aufgeloeste Punkte/Konturen) nachgefuehrt, Metriken eines Teils uebernehmbar (USE_MY_METRICS samt hmtx). 19.3 name: Macintosh-Eintrag in Mac OS Roman (Fonts/MacOsRoman nach Apples ROMAN.TXT), gesetzt wird je Sprache (andere Sprachen bleiben), ReadRecords. 19.4 OS/2: fsSelection, Unicode-/Codepage-Bereiche, achVendID, xAvgCharWidth lesen/setzen (mittlere Breite nach der Regel ab v3 berechenbar), Hoehen gegen die Umrisse geprueft (CheckHeights). 19.5 senkrechte Metriken vhea/vmtx setzbar, fehlende Tabellen angelegt. 19.6 Kerning-Klassen wirken rueckwirkend (Arbeitsstand fuehrt Klassen und Klassenpaare, Einzelpaar vor Klassenpaar). Alles in Kern, MCP, REST und Schrift-Erzeuger; im Browser belegt (Sprachwahl, Herstellerkennung, Hoehenbefund). Ehrlich: die alte Angabe „loest bis Tiefe 5 auf" war falsch - gemessen liest GlyphBoundsReader bis Tiefe 10; das Punkt-Andocken steht als FCR22 offen (gespiegelte Leser); in der Datei stehen Kerning-Klassen weiterhin als Einzelpaare (die GPOS-Klassenform schreibt der Kern nicht), eine neu geoeffnete Schrift kennt die Klassen nicht mehr; Unicode-Bereiche werden gesetzt, nicht aus der cmap berechnet	M

#	Punkt	Stand	Aufw.
FCR22	Zusammengesetzte Glyphen mit Punkt-Andocken lesen	M1509 - der Teil, dessen Argumente Punktnummern sind (ARGS_ARE_XY_VALUES nicht gesetzt), wird so versetzt, dass Punkt arg2 des neuen Teils (nach seiner 2x2-Abbildung) auf Punkt arg1 der bisher gesetzten Teile faellt: im Zeichen-/Grenzenweg (GlyphBoundsReader.ResolveComposite, die Punktdekodierung ist dafuer aus dem Zeichenweg herausgezogen, kein zweiter Leser), in TrueTypeFont.CompositeParts (ueber dieselbe Aufloesung) und beim Instanziieren variabler Schriften fuer die Grenzen (VariableFont.VisitPoints). Eine Punktnummer ausserhalb gibt Versatz 0/0 statt einer Ausnahme. Gespiegelt nach Kuonen.UI (TrueTypeFont.cs, GlyphBoundsReader.cs, eigener MSTest dort, 425 Text-Tests gruen). Ehrlich: der Weg der variablen Schriften ist ohne eigenen Andock-Test (nur gegen die vorhandenen Varianten-Tests geprueft) - es gibt keine variable Fixture mit angedockter Komponente	M
FCR18	Restposten in der Probe	M1510 - neu im Kern Fonts/FontShaping (oeffentliche Formung ueber den vorhandenen TextShaper: Ligaturen/Ersetzungen aus GSUB, Unterschneidung aus GPOS bzw. kern, komplexe Schriften, visuelle Reihenfolge). Die Probe im Schrift-Erzeuger setzt auf Wunsch geformt an der gebauten Schrift des Arbeitsstands (offene Aenderungen inklusive), bricht einen 12-pt-Absatz auf eine Breite um (gierig an Leerzeichen, ein zu breites Wort bekommt eine eigene Zeile) und zeigt neben jeder Zeile das Bildschirmbild in echter Pixelgroesse (1 pt = 1 px, GlyphRasterizer, Hinting schaltbar). MCP font_info mit text gibt die geformte Glyphenfolge aus. Im Browser belegt (9 Bildschirmbilder im Wasserfall, 3 Zeilen bei 50 pt Umbruch). Ehrlich: keine Silbentrennung im Umbruch; das Bildschirmbild setzt Glyphen auf ganze Pixel ohne Subpixel-Positionierung und ohne ClearType-Farbsaum	M

#	Punkt	Stand	Aufw.
FCR13	Variable Schriften bearbeiten	M1511 - Kern: die benannte Grenze „keine Eck-Master“ ist geschlossen: neu Fonts/VariationModel (Reihenfolge und Wirkungszonen des allgemeinen Variationsmodells, wie fontTools.varLib.models), EINE Stelle fuer VariableFontBuilder (gvar) und Cff2Writer (CFF2) - vorher stand die Rechnung zweimal da. Gegen fontTools 4.63 (scripts/eckmaster-orakel.cs): 35 Lagen eines zweiachsigen Raums mit Eck-Schnitt Punkt fuer Punkt gleich. Cff2Writer schreibt jetzt benannte Instanzen, Achsnamen, STAT und avar (dieselben Designraum-Tabellen wie gvar). Neu VariableFontDecomposer: eine bestehende variable Schrift zerfaellt in ihre Schnitte (die Spitzen aus gvar bzw. dem CFF2-VarStore, ueber die umgekehrte avar-Karte in Nutzerwerte) und baut sich gleich wieder - auch die fremd gebauten Fixtures (glyf und CFF2) am Gitter. VariableFont.Instantiate nimmt auch CFF2. MCP/REST: font_edit//font-edit op: instantiate und build_variable (Achsen, Schnitte, Instanzen, avar), font_info nennt Achsnamen, Instanzen und die Lagen der Schnitte. Schrift-Erzeuger: Reiter „Variation“ - Achsen anlegen/entfernen, Regler je Achse, Schnitte anlegen (Kopie des Stands im Editor) und einzeln bearbeiten (der Editor haelt immer EINEN festen Schnitt, die anderen liegen in ihren Faechern), benannte Instanzen, Probe am Regler zeigt den Zwischenschnitt live; eine geoeffnete variable Schrift wird zerlegt, Speichern baut sie neu. Im Browser belegt (Kante 300/400/500 bei wght 0/500/1000, Zwischenschnitt bei 500 und Instanz „Halbfett“ gespeichert und wieder geladen). Ehrlich: Kerning, Namen und OS/2 kommen aus dem Standard-Schnitt (variable GPOS/HVAR/MVAR werden nicht geschrieben); eine Vorlage mit von Hand gesetzten Wirkungszonen, die das Modell nicht ergibt, trifft der Neubau an den Schnitten, dazwischen nach dem Modell	L
FCR14	Farbschriften bearbeiten	M1512 - Kern: neu Fonts/ColorFontEditor + ColorFontDesign: der Entwurf einer Farbschrift (ALLE Paletten, COLR-v0-Ebenen roh mit Palettenindex, COLR-v1-Ebenen mit Fuellung/Transformation/Mischmodus, SVG-Dokumente, Bitmap-Stufen) lesen und ganz schreiben (fehlende Teile verschwinden). ColorFontEncoder.EncodeWithPalettes (mehrere Paletten, hell/dunkel), Bitmap-Schreiber mit beliebig vielen Stufen (ColorBitmapStrike), ColorBitmapGlyph.Advance. Behoben: der CBDT-Index schrieb nur je BELEGTER Glyphe einen Versatz - bei Luecken las der Leser falsche Bilder (fontTools 4.63 bestaetigt jetzt beide Tabellen, scripts/farbbitmap-orakel.cs). MCP/REST: font_edit//font-edit op: color (Paletten, Ebenen v0/v1 mit Verlaeufen, SVG, Bitmaps, clear), font_info nennt den Farb-Entwurf. Schrift-Erzeuger: Reiter „Farbe“ - Paletten mit Farbfeld und Deckkraft, Palette als Kopie anlegen, Ebenen der aktiven Glyphe (Palettenebenen und Verlaufsebenen mit Geometrie, Farbstopps, Matrix, Mischmodus, verschieben/entfernen), Vorschau mit echten SVG-Verlaeufen, SVG-Glyphe (Text oder Datei), Bitmap-Glyphe (PNG in Stufe, CBDT oder sbix). Im Browser belegt (Palettenfarbe, Verlaufsebene, Speichern und Neuladen, Bild in 64 ppem). Ehrlich: sobald eine Glyphe v1-Ebenen traegt, wird die ganze COLR als v1 geschrieben und weitere Paletten gehen verloren (v1-Fuellungen tragen ihre Farbe selbst); ein Kegelverlauf erscheint in der Vorschau einfarbig (SVG kennt ihn nicht)	L

#	Punkt	Stand	Aufw.
LVS25.1	vsdx_check (g) VSDX_ZEICHENLAUF_REIHENFOLGE — ein <cp>/<pp> springt auf eine schon vor einem anderen Lauf benutzte Zeile zurueck (0,0,1,0); Visio setzt den Text dann Zeichen fuer Zeichen uebereinander. Ursache im eigenen Schreiber: gleiche Formate teilten sich EINE Zeile, auch nach einem anderen Format (model_to_vsdx mit farbigen Abschnitten war betroffen); jetzt nur direkt hintereinander	M1501	S
LVS25.2	Beziehungsenden je Relation (begin_end/end_end: none/one/only_one/zero_or_one/many/zero_or_many/one_or_many = Visio 0/24/25/30/27/29/28), end_size 0..6, Linienname name	M1501	S
LVS25.3	attach:"edges" — Unterkante der Quelle (Startpunkte gleichmaessig verteilt, nach der Lage der Ziele) zur Mitte der Oberkante des Ziels, beidseits an echten Verbindungspunkten verklebt; der Schreiber klebt dazu an JEDEN Verbindungspunkt ueber seine Zeile (Connections.X{n}, ToPart 100+n-1), nicht nur an X1	M1501	S
LVS25.4	Klebung an die Gruppe bei group:true — war seit M1467 erfuellt, jetzt mit Test belegt (Connect und _XFTRIGGER zeigen auf die Gruppe)	M1501	S
LVS25.5	Titeltext-Format je Entitaet: header_text_color, header_bold	M1501	S
LVS25.6	body:false ohne Rest (nur titelhoch, Titelfarbe, kein Rumpftext) — war erfuellt, jetzt mit Test belegt	M1501	S
LVS25.7	Freie Textfelder texts (x_mm/y_mm von oben, width_mm, size_pt, color, bold, align); sie bestimmen die Seitengroesse mit	M1501	S
LVS25.8	Rahmen ohne Fuellung frames (zuunterst gezeichnet)	M1501	S
LVS25.9	layout:"tree" mit wrap:false (Kinder immer in einer Reihe darunter, Elternteil mittig ueber ihnen, Teilbaumbreiten, Ringschluss-sicher) und page:{fit:"content", min, margin_mm} (A0–A6, Letter, Legal, -landscape)	M1501	S
LVS25.10	Datenquelle (role:"source") mittig ueber den Wurzeln, side_text rechts daneben	M1501	S
LVS25.11	behind:false legt die Linie auch bei attach:"center" ueber die Kaesten — war erfuellt, jetzt mit Test belegt	M1501	S
LVS25.12	Ansichtsschalter als Kindelemente von <Window> — war erfuellt, jetzt mit Test belegt	M1501	S
LVS25.13	vsdx_describe: wirksame Zellwerte (Fuell-/Linienfarbe, Strich, Muster, Pfeile samt Groesse, Schrift/Groesse/Farbe/fett/kursiv)	M1501	S
LVS25.14	vsdx_describe: Klebungen je Seite (connects) und je Linie	M1501	S

#	Punkt	Stand	Aufw.
LVS25.15	Gesperrte Dateien lesen: alle MCP-Werkzeuge lesen mit FileShare.ReadWrite|Delete (eine in Visio/OneDrive geoeffnete Datei)	M1501	S
LVS25.16	vsdx_describe: vom Master geerbte Zellen aufgeloest (der Leser verschmilzt sie bereits; mit Test belegt)	M1501	S
LVS25.17	vsdx_check (h) VSDX_KLEBUNG_AN_GRUPPENMITGLIED (Connect oder _XFTRIGGER auf ein Gruppenmitglied) und (i) VSDX_LINIE_FREI (Verbinder Warnung, gewoehnliche Linie Hinweis)	M1501	S
LVS25.18	vsdx_to_pdf setzt fett mit der Ersatzschrift fett — nicht nachvollziehbar: in allen nachgestellten Faellen (eigene Zeile, Gruppe, Formatvorlage, Formvorlage, Schrift als Nummer) war es fett; mit Test belegt, die konkrete Datei des Nutzers steht aus	M1501	S
LVS25.19	Linien folgen in Visio beim Verschieben (Nutzermeldung „Linien bleiben am alten Ort liegen / gehen beim Aendern verloren“): die Geometrie einer verklebten Linie stand als feste Zahl (Begin/End wanderten, der Strich blieb), jetzt als Formel auf Width/Height wie Visios eigene Linie; Verbindungspunkte als Bruchteil der Formgroesse	M1501	S

#	Vorbild	Punkt	Stand	Aufw.
ED43b	Word	Restposten des Fliesstext-Modus (nach ED43a)	M1502 - Loeschen des letzten ausgezeichneten Zeichens nimmt die Marken mit, Eingabe in einer Liste beginnt den naechsten Punkt (ED43b.1); Zwischenablage mit Formatierung (ED43b.2); nicht jede Taste baut alle Seitenbilder (~1000 ms -> 30-57 ms je Taste, ED43b.3); Schreibstrich in Tabellenzellen, Bildunterschriften, Formeln (als LaTeX-Quelltext) und im Inhaltsverzeichnis (als Sprung, ED43b.4); Fliesstext zwischen festen Seiten (FlowRegion.InsertBeforePage), Seitenzahlen/Querverweise an der eingeordneten Stelle, Verzeichnis im Fixpunkt, Notentexte am Seitenfuss (PageFootnotes) - auch nach dem .kdoc-Rundlauf (ED43b.5); der uebernommene Fliesstext auf der Rahmen-Leinwand: Fluss-Seiten in der Seitenleiste, eigene Ansicht mit dem Rahmen als Objekt (ziehen, Groesse, mm-Felder, Loeschen, rueckgaengig; SetBounds/RemoveFlow/RemoveColumns/RenderedPageOf), Vorschau/Hintergrund/Miniaturen zeigen die richtige PDF-Seite (ED43b.6). Ehrliche Grenzen: ein uebernommener Fliesstext laesst sich nicht in den Fliesstext-Modus zurueckholen (die Uebernahme setzt ihn in Layout-Objekte; bearbeitet wird er als Rahmen bzw. ueber neue Uebernahme); Strukturbaum-Reihenfolge und gespiegelte Raender folgen der Anlage-, nicht der eingeordneten Reihenfolge	L