Legacy PDF to XML Pipeline Documentation

Overview

This document describes the end-to-end process of `pdf_to_unified_xml.py`, the legacy (non-AI) PDF to XML conversion pipeline.

Programs (Python Files) Used

Order	Python File	Purpose
1	`pdf_to_excel_columns.py`	Text extraction with column detection & reading order
2	`Multipage_Image_Extractor.py`	Media extraction (images, tables, vectors)
3	`enhanced_word_split_fixer.py`	Fixes split words from PDF extraction
4	`reference_mapper.py`	Tracks image transformations (optional)
5	`font_roles_auto.py`	Auto-derives font roles from unified XML
6	`heuristics_Nov3.py`	Applies heuristics to create structured DocBook
7	`create_book_package.py`	Packages DocBook into deliverable ZIP
8	`rittdoc_compliance_pipeline.py`	DTD validation & compliance fixes
9	`editor_server.py`	Web-based UI editor (optional, with `--edit-mode`)

Pipeline Flow Diagram

```
INPUT: document.pdf
         |
         v
+-------------------------------------------------------------------+
|  PHASE 1: Unified XML Creation                                    |
+--------------------------------------------------------------------+
|                                                                    |
|  Step 1: extract_table_bboxes_fast()                               |
|          +-- Detects table regions (for exclusion)                 |
|                                                                    |
|  Step 2: pdf_to_excel_with_columns()                               |
|          |-- Input: PDF + table exclusion regions                  |
|          |-- Uses: pdftohtml -> intermediate XML                   |
|          +-- Output: {base}_columns.xlsx  (debug)                  |
|                      pdftohtml.xml (temp)                          |
|                                                                    |
|  Step 3: extract_media_and_tables()                                |
|          |-- Input: PDF                                            |
|          |-- Uses: PyMuPDF (fitz), Camelot                         |
|          +-- Output: {base}_media.xml                              |
|                      {base}_MultiMedia/  (folder with images)      |
|                                                                    |
|  Step 4: parse_media_xml()                                         |
|          +-- Parses media XML into memory structures               |
|                                                                    |
|  Step 5: merge_text_and_media_simple()                             |
|          +-- Removes text fragments inside tables/media            |
|                                                                    |
|  Step 6: create_unified_xml()                                      |
|          |-- Creates paragraphs from text fragments                |
|          |-- Merges media into document structure                  |
|          +-- Output: {base}_unified.xml  <-- MAIN OUTPUT           |
|                      {base}_reference_mapping_phase1.json          |
|                                                                    |
+--------------------------------------------------------------------+
         |
         v  (with --full-pipeline flag)
+--------------------------------------------------------------------+
|  PHASE 2: DocBook Processing                                       |
+--------------------------------------------------------------------+
|                                                                    |
|  Step 6: font_roles_auto.py                                        |
|          |-- Input: {base}_unified.xml                             |
|          +-- Output: {base}_font_roles.json                        |
|                                                                    |
|  Step 7: heuristics_Nov3.py                                        |
|          |-- Input: unified XML + font_roles JSON                  |
|          |-- Detects: headers, footers, TOC, chapters              |
|          +-- Output: {base}_structured.xml                         |
|                                                                    |
|  Step 8: create_book_package.py                                    |
|          |-- Input: structured XML + metadata                      |
|          |-- Creates: DocBook 4.2 package                          |
|          +-- Output: {base}_package/pre_fixes_{isbn}.zip           |
|                                                                    |
+--------------------------------------------------------------------+
         |
         v  (unless --skip-validation)
+--------------------------------------------------------------------+
|  PHASE 3: RittDoc Validation & Compliance                          |
+--------------------------------------------------------------------+
|                                                                    |
|  Step 9: RittDocCompliancePipeline                                |
|          |-- Input: pre_fixes_{isbn}.zip                          |
|          |-- Validates against RittDoc DTD                        |
|          |-- Auto-fixes DTD violations (up to 3 iterations)       |
|          +-- Output: {isbn}.zip  (compliant)                      |
|                      {isbn}_validation_report.xlsx                |
|                                                                   |
+-------------------------------------------------------------------+
         |
         v  (with --edit-mode)
+-------------------------------------------------------------------+
|  OPTIONAL: Web Editor                                             |
+-------------------------------------------------------------------+
|                                                                   |
|  editor_server.py                                                 |
|          +-- Flask-based web UI for manual corrections            |
|                                                                   |
+-------------------------------------------------------------------+
```

Summary of Intermediate Outputs

File/Folder	Created By	Description
`{base}_columns.xlsx`	`pdf_to_excel_columns.py`	Debug: text with coordinates
`pdftohtml.xml`	`pdf_to_excel_columns.py`	Temp: raw pdftohtml output
`{base}_media.xml`	`Multipage_Image_Extractor.py`	Media coordinates & metadata
`{base}_MultiMedia/`	`Multipage_Image_Extractor.py`	Extracted images folder
`{base}_unified.xml`	`create_unified_xml()`	**Phase 1 output**: merged text + media
`{base}_reference_mapping_phase1.json`	`reference_mapper.py`	Image transformation tracking
`{base}_font_roles.json`	`font_roles_auto.py`	Font to semantic role mapping
`{base}_structured.xml`	`heuristics_Nov3.py`	DocBook with chapters/sections

File/Folder	Created By	Description
`pre_fixes_{isbn}.zip`	`create_book_package.py`	DocBook ZIP (pre-validation)
`{isbn}.zip`	`rittdoc_compliance_pipeline.py`	**Final output**: RittDoc compliant
`{isbn}_validation_report.xlsx`	`rittdoc_compliance_pipeline.py`	Validation results

Key Libraries Used (Non-AI)

Library	Purpose
pdftohtml	Converts PDF to intermediate XML
PyMuPDF (fitz)	Text extraction, image rendering
Camelot-py	Table detection
lxml	XML manipulation
pdfplumber	Layout analysis

Usage Examples

Basic Usage (Phase 1 only - unified XML)

```
python pdf_to_unified_xml.py document.pdf
```

Full Pipeline with Validation

```
python pdf_to_unified_xml.py document.pdf --full-pipeline
```

Full Pipeline without Validation

```
python pdf_to_unified_xml.py document.pdf --full-pipeline --skip-validation
```

With Web Editor

```
python pdf_to_unified_xml.py document.pdf --full-pipeline --edit-mode
```

Command Line Options

Option	Description
`--dpi`	DPI for image rendering (default: 200)
`--out`	Optional output directory (default: same as PDF)
`--full-pipeline`	Run full DocBook processing pipeline
`--skip-packaging`	Skip final ZIP packaging step
`--metadata-dir`	Directory containing metadata.csv or metadata.xls/xlsx
`--dtd`	Path to DTD file for validation
`--skip-validation`	Skip RittDoc validation step
`--no-caption-filter`	Include all detected tables, even without 'Table X' captions
`--caption-distance`	Maximum distance between table and caption (default: 100.0)
`--edit-mode`	Launch web-based UI editor after creating unified XML
`--editor-port`	Port for editor server (default: 5555)

Phase Details

Phase 1: Unified XML Creation

1. **Table Detection**: Detects table bounding boxes to exclude from text processing
2. **Text Extraction**: Uses `pdftohtml` to extract text with coordinates, then applies column detection and reading order algorithms
3. **Media Extraction**: Extracts images, tables, and vector graphics using PyMuPDF and Camelot
4. **Overlap Removal**: Filters out text fragments that overlap with detected tables/media
5. **Unified XML**: Merges text and media into a single hierarchical XML with page number IDs

Phase 2: DocBook Processing

1. **Font Role Analysis**: Analyzes font sizes and styles to derive semantic roles (headings, body text, etc.)
2. **Heuristic Structuring**: Applies rules to detect headers, footers, TOC, chapters, and other structural elements
3. **DocBook Packaging**: Creates a DocBook 4.2 compliant ZIP package with all assets

Phase 3: RittDoc Validation

1. **DTD Validation**: Validates the package against the RittDoc DTD
2. **Auto-Fix**: Automatically fixes common DTD violations (up to 3 iterations)
3. **Compliance Report**: Generates an Excel report with validation results

---

Architecture Comparison: Legacy vs Modern

Aspect	Legacy Pipeline	Modern AI Pipeline
Entry Point	`pdf_to_unified_xml.py`	`pdf_orchestrator.py`
Text Extraction	pdftohtml + heuristics	Claude Vision API
Layout Detection	Column clustering algorithms	AI visual understanding
Structure Detection	Font analysis + rules	AI semantic analysis
Output Format	DocBook 4.2 XML	DocBook 4.2 XML
Validation	RittDoc DTD	RittDoc DTD

Generated from codebase analysis of PDFtoXMLUsingExcel legacy pipeline