**Power Apps → Power Automate (Cloud) → PAD → Python → SharePoint  
End‑to‑End Implementation Guide**

Rocco Rossinni  
Date: 2025-10-07

# Preface & scope

This document explains how to wire existing Canvas app to a cloud flow that calls a Power Automate for desktop (PAD) flow to run local Python script (merge\_pdfs.py), then store the merged bundle(s) in SharePoint and return share links back to Power Apps.

# Simple definitions (what these terms mean)

**Tenant:** Exelixis Microsoft 365 environment—basically organization's private space in Microsoft Cloud. It contains users, groups, SharePoint sites, OneDrive storage, and Power Platform environment.

**Power Apps (Canvas app):** The app screen built where the user clicks a button to request the merged PDF(s).

**Power Automate cloud flow:** An online workflow that starts when Power Apps calls it. It can call a desktop flow, write files to SharePoint, and return data back to Power Apps.

**PAD (Power Automate for desktop):** Automates things on a Windows computer. Here it runs Python script on that machine, optionally unattended (license permitting).

**SharePoint library:** A folder-like place in SharePoint Online where files live. When synced with OneDrive, it appears as a normal folder in File Explorer.

**OneDrive/SharePoint Sync (local sync root):** The feature that makes a SharePoint library appear as a local Windows folder. Example local path: C:\Users\<you>\Shared Documents\…

# Key requirements (from spec)

* Create a single merged bundle when either Tables or Figures is chosen.
* If neither is chosen, auto‑split into two bundles (tables and figures).
* Preserve the original file order from the source folder.
* Do not strip naming prefixes (t\_/f\_ or t‑/f‑).
* Support optional hyperlinked TOC via --toc.

# Step 1 — Canvas app (scrPdfToc)

Add the following input controls (Modern controls recommended):

* Text input: txtSourcePath (Value holds a path like /Documents/myStudy/src)
* Text input: txtOutputFolder (Value holds a path like /Documents/myStudy/out)
* Toggle: tglTables (Default true/false as you prefer)
* Toggle: tglFigures
* Toggle: tglToc
* Dropdown: drpPageFormat (Items: ["letter", "A4"])

Set the button's OnSelect formula to call the flow and handle results:

// Button OnSelect  
UpdateContext({ locBusy: true });  
Set(  
 varResult,  
 Flow\_AddPdfToc\_Cloud.Run({  
 sourcePath: Trim(txtSourcePath.Value),  
 outputFolder: Trim(txtOutputFolder.Value),  
 tables: tglTables.Value,  
 figures: tglFigures.Value,  
 toc: tglToc.Value,  
 pageFormat: drpPageFormat.Selected.Value  
 })  
);  
If(  
 Lower(Coalesce(varResult.status, "")) = "ok",  
 Notify("TOC bundle created", NotificationType.Success),  
 Notify("Error: " & Coalesce(varResult.error, "Unknown error"), NotificationType.Error)  
);  
UpdateContext({ locBusy: false });

# Step 2 — Cloud flow: Flow\_AddPdfToc\_Cloud

1. Trigger: Power Apps (V2). Create these inputs (Ask in Power Apps): sourcePath (Text), outputFolder (Text), tables (Yes/No), figures (Yes/No), toc (Yes/No), pageFormat (Text).
2. Optional: Validate flags. If both tables and figures are true, either call twice or clear both to allow auto‑split (no flags).
3. Action: Run a flow built with Power Automate for desktop (your PAD flow). Pass all six inputs to PAD.
4. If the library is synced via OneDrive: PAD will write directly into the synced folder; the file will appear in SharePoint after sync. Then use 'Create sharing link' in SharePoint connector and capture the URL(s).
5. If not synced: Have PAD return file content or place the files in a temp path and have the cloud flow upload via 'Create file' (SharePoint), then 'Create sharing link'.
6. Respond to Power Apps with JSON: status, outputPath (or outputPaths array), shareLink (or shareLinks array), error.

# Step 3 — PAD desktop flow (runs on a Windows machine)

Create a PAD flow with these inputs and outputs:

* Inputs (Text/Boolean): sourcePath, outputFolder, tables, figures, toc, pageFormat.
* Outputs (Text/List/Text): status, outLocalPaths (list), stdout, stderr, error.

High‑level actions inside the PAD flow:

1. Translate SharePoint site‑relative paths (e.g., /Documents/myStudy/src) to local synced paths (e.g., C:\Users\<you>\Contoso\Shared Documents\myStudy\src).
2. Build the Python command string: base script path, -s source, -o output, --page-format, plus flags:
3. • If toc = true → add --toc
4. • If tables = true and figures = false → add --tables
5. • If figures = true and tables = false → add --figures
6. • If both false → no flag (auto‑split into two bundles)
7. Run application (or Run DOS command) to execute Python. Capture stdout and stderr.
8. Parse stdout lines that begin with 'Created:' to collect the fullpaths of the outputs.
9. Set status = 'OK' if at least one 'Created:' path is found; else 'ERROR' and include stderr.
10. Return outLocalPaths and any captured logs back to the cloud flow.

Command examples you can paste into PAD's 'Run application':

Tables only with TOC (letter):  
python "C:\apps\pdfbundle\merge\_pdfs.py" -s "C:\SP\Shared Documents\myStudy\src" -o "C:\SP\Shared Documents\myStudy\out\\_merged.pdf" --toc --tables --page-format letter  
  
Auto‑split both bundles with TOC (A4):  
python "C:\apps\pdfbundle\merge\_pdfs.py" -s "C:\SP\Shared Documents\myStudy\src" -o "C:\SP\Shared Documents\myStudy\out\\_merged.pdf" --toc --page-format A4

# Screenshot capture checklist (what to show and how to capture)

Use \*\*Win + Shift + S\*\* → Rectangle selection to capture. Save each PNG and insert into the final handoff doc. Redact any sensitive names or paths if needed.

**1) Canvas inputs/button**

* Open Power Apps Studio and your Canvas app screen (scrPdfToc).
* Ensure txtSourcePath, txtOutputFolder, tglTables, tglFigures, tglToc, drpPageFormat, and the action button are visible.
* Include the button's OnSelect formula pane if possible (collapsed or expanded).
* Filename suggestion: 01\_canvas\_inputs\_button.png

A screenshot of a computer

AI-generated content may be incorrect.

**2) Cloud flow trigger + inputs**

* Open Power Automate → your solution → the flow: Flow\_AddPdfToc\_Cloud.
* Show the 'Power Apps (V2)' trigger with all Ask in Power Apps parameters defined.
* Filename: 02\_flow\_trigger\_inputs.png

**3) Run desktop flow action**

* In the same cloud flow, scroll to the 'Run a flow built with Power Automate for desktop' action.
* Show the mapping of all six parameters to the PAD flow inputs.
* Filename: 03\_flow\_run\_desktop\_action.png

**4) PAD variables**

* Open PAD designer on the Windows machine.
* Select your desktop flow and show the input/output variables pane.
* Filename: 04\_pad\_variables.png

**5) PAD Run Application action**

* In PAD, open the step that runs Python (Run application / Run DOS command).
* Ensure the full command string is visible (source, output, flags).
* Filename: 05\_pad\_run\_application.png

**6) PAD outputs**

* Show the step that parses stdout and the variables where outLocalPaths/status are set.
* Filename: 06\_pad\_outputs.png

**7) SharePoint folder with outputs**

* In SharePoint (web) or File Explorer (synced), open the target folder (e.g., /Documents/myStudy/out).
* Show the created files (\_merged\_tables.pdf and/or \_merged\_figures.pdf).
* Filename: 07\_sharepoint\_outputs.png

A screenshot of a computer

AI-generated content may be incorrect.

**8) Opening the share link from the app**

* Run the Canvas app and click the link button (Launch).
* Capture the PDF open in the browser (redact identifiers if needed).
* Filename: 08\_open\_share\_link.png

# Functional validation (what to test)

* Tables only: one merged PDF with 'tables' set, order preserved, names intact.
* Figures only: one merged PDF with 'figures' set, order preserved, names intact.
* Auto‑split: with neither tables nor figures selected, both bundles appear.
* TOC: when toc = true, the output has a hyperlinked table of contents.
* Links: the returned share link(s) open from Power Apps.

# Troubleshooting

* In Power Apps, Modern text inputs expose .Value (not .Text) — ensure formulas use .Value.
* Do not pass both --tables and --figures in one call. Either call twice or rely on auto‑split.
* Large PDFs: prefer writing directly into the synced folder so the cloud flow only creates links.
* If nothing is created, check PAD's stdout/stderr and verify 'Created:' lines from Python.

# Appendix A — Power Fx snippets

// Launch the first link if present  
If(  
 !IsBlank(varResult.shareLink),  
 Launch(varResult.shareLink),  
 If(!IsBlank(varResult.shareLinks), Launch(First(varResult.shareLinks).Value))  
)  
  
// Show all output paths in a label  
If(  
 IsBlank(varResult.outputPaths),  
 varResult.outputPath,  
 Concat(varResult.outputPaths, Value & Char(10))  
)

# Appendix B — Cloud flow 'Respond to Power Apps' schema (example)

{  
 "status": "OK",  
 "outputPaths": ["/Documents/myStudy/out/\_merged\_tables.pdf", "/Documents/myStudy/out/\_merged\_figures.pdf"],  
 "shareLinks": ["https://contoso.sharepoint.com/...tables.pdf", "https://contoso.sharepoint.com/...figures.pdf"],  
 "error": ""  
}

# Appendix C — Self‑hosted API alternative (FastAPI or Flask)

If you prefer not to use PAD, you can host your Python merging logic behind a local API. Two minimal patterns are below. Both accept JSON with source/output paths/flags and return either base64 PDF(s) or write to SharePoint and return links. Enable CORS so Power Apps or your cloud flow can call it.

FastAPI — minimal endpoint:  
# FastAPI example (uvicorn app:main --reload)  
from fastapi import FastAPI, HTTPException  
from pydantic import BaseModel  
from typing import List, Optional  
import subprocess, json, tempfile, os, base64  
  
app = FastAPI(title='PDF Merge API')  
  
class MergeRequest(BaseModel):  
 sourcePath: str  
 outputFolder: str  
 tables: bool = False  
 figures: bool = False  
 toc: bool = True  
 pageFormat: str = 'letter'  
  
@app.post('/merge')  
def merge(req: MergeRequest):  
 # Build command  
 cmd = ['python', 'C:/apps/pdfbundle/merge\_pdfs.py', '-s', req.sourcePath, '-o', os.path.join(req.outputFolder, '\_merged.pdf'), '--page-format', req.pageFormat]  
 if req.toc: cmd.append('--toc')  
 if req.tables and not req.figures: cmd.append('--tables')  
 if req.figures and not req.tables: cmd.append('--figures')  
 # Run and capture  
 p = subprocess.run(cmd, capture\_output=True, text=True)  
 if p.returncode != 0:  
 raise HTTPException(status\_code=500, detail=p.stderr)  
 # Parse outputs  
 paths = [line.split('Created:',1)[1].strip() for line in p.stdout.splitlines() if line.startswith('Created:')]  
 # Option A: return paths only  
 return {'status':'OK','outputPaths':paths}

Flask — minimal endpoint:  
# Flask example (set FLASK\_APP=app.py; flask run --host 0.0.0.0 --port 5000)  
from flask import Flask, request, jsonify  
import subprocess, os  
  
app = Flask(\_\_name\_\_)  
  
@app.post('/merge')  
def merge():  
 data = request.get\_json(force=True)  
 cmd = ['python', 'C:/apps/pdfbundle/merge\_pdfs.py', '-s', data['sourcePath'], '-o', os.path.join(data['outputFolder'], '\_merged.pdf'), '--page-format', data.get('pageFormat','letter')]  
 if data.get('toc'): cmd.append('--toc')  
 if data.get('tables') and not data.get('figures'): cmd.append('--tables')  
 if data.get('figures') and not data.get('tables'): cmd.append('--figures')  
 p = subprocess.run(cmd, capture\_output=True, text=True)  
 if p.returncode != 0:  
 return jsonify({'status':'ERROR','stderr':p.stderr}), 500  
 paths = [line.split('Created:',1)[1].strip() for line in p.stdout.splitlines() if line.startswith('Created:')]  
 return jsonify({'status':'OK','outputPaths':paths})

Hosting & security notes:

* Run behind your firewall on a Windows server or VM (so the script can access local/synced folders).
* Enable CORS for Power Apps origins (or call it only from your cloud flow).
* Protect with network ACLs or an API key/Basic auth (reverse proxy like IIS/ARR or nginx).
* Service options: run as a Windows Service (e.g., NSSM) or a scheduled task that starts uvicorn/gunicorn on boot.

# Appendix D — Glossary (quick reference)

**Tenant:** Your company's Microsoft 365 environment—basically your organization's private space in Microsoft Cloud. It contains your users, groups, SharePoint sites, OneDrive storage, and Power Platform environment.

**Power Apps (Canvas app):** The app screen you built where the user clicks a button to request the merged PDF(s).

**Power Automate cloud flow:** An online workflow that starts when Power Apps calls it. It can call a desktop flow, write files to SharePoint, and return data back to Power Apps.

**PAD (Power Automate for desktop):** Automates things on a Windows computer. Here it runs your Python script on that machine, optionally unattended (license permitting).

**SharePoint library:** A folder-like place in SharePoint Online where files live. When synced with OneDrive, it appears as a normal folder in File Explorer.

**OneDrive/SharePoint Sync (local sync root):** The feature that makes a SharePoint library appear as a local Windows folder. Example local path: C:\Users\<you>\Contoso\Shared Documents\…