Please review **ready-to-deploy zipped Azure Function package** for adding a hyperlinked TOC to PDFs, along with supporting assets:

**"\global\macros\star\tracker\_dev\biometrics\_tracker\_build\pdf\_toc\_function\_package"**

**Included Files:**

* **Azure Function App**
  + function\_app.py: Python v2 HTTP trigger using PyMuPDF
  + requirements.txt: Dependencies
  + host.json: Azure Functions config
  + README.md: Deployment and usage instructions
* **Flow Build Guide**
  + flow\_build\_guide.txt: Step-by-step instructions for building Flow\_AddPdfToc\_Cloud
* **Power Apps One-Pager**
  + power\_apps\_one\_pager.txt: Screen wiring and formulas for scrPdfToc

**1) Azure Function (Python) — “Add PDF TOC”**

This is the serverless piece the flow calls. It receives the **PDF as Base64**, inserts TOC pages based on **existing PDF bookmarks**, and returns the **modified PDF as Base64**.

Use the **Python v2 programming model** (decorators) and deploy on an **Azure Functions (Linux) plan**. [[3]](https://learn.microsoft.com/en-us/azure/azure-functions/functions-reference-python)

**function\_app.py**

# Azure Functions (Python v2) - HTTP Trigger

import logging, json, base64

import azure.functions as func

import fitz # PyMuPDF

app = func.FunctionApp(http\_auth\_level=func.AuthLevel.FUNCTION)

@app.function\_name(name="AddPdfToc")

@app.route(route="add-pdf-toc", methods=["POST"]) # POST https://&lt;app&gt;.azurewebsites.net/api/add-pdf-toc?code=&lt;function\_key&gt;

def add\_pdf\_toc(req: func.HttpRequest) -&gt; func.HttpResponse:

try:

body = req.get\_json()

pdf\_b64 = body["fileContent"]

title = body.get("title", "Hyperlinked Table of Contents")

zoom = float(body.get("zoom", 1.0))

pdf\_bytes = base64.b64decode(pdf\_b64)

doc = fitz.open(stream=pdf\_bytes, filetype="pdf")

try:

toc = doc.get\_toc(simple=False)

if not toc:

toc = [[lvl, t, p, {}] for (lvl, t, p) in doc.get\_toc()]

if not toc:

return func.HttpResponse(

json.dumps({"status": "ERROR", "error": "No bookmarks found in PDF. Create bookmarks first."}),

status\_code=400, mimetype="application/json")

LEFT, RIGHT, TOP = 54, 54, 54

LINE\_H, FS = 18, 11

FONT, COLOR = "helv", (0, 0, 1)

first\_rect = doc[0].rect

lines\_per = max(1, int(((first\_rect.height - 2 \* TOP)//LINE\_H) - 2))

num\_toc\_pages = (len(toc) + lines\_per - 1)//lines\_per

for \_ in range(num\_toc\_pages):

doc.new\_page(pno=0)

page\_index = 0

cur = doc[page\_index]

y = TOP

cur.insert\_text((LEFT, y), title, fontsize=16, fontname=FONT, fill=(0,0,0))

y += 2\*LINE\_H

for (lvl, title\_i, p1, \_meta) in toc:

if y &gt; cur.rect.height - TOP:

page\_index += 1

cur = doc[page\_index]

y = TOP

cur.insert\_text((LEFT, y), f"{title} (cont.)", fontsize=14, fontname=FONT, fill=(0,0,0))

y += 2\*LINE\_H

x = LEFT + max(0, (int(lvl)-1))\*14

label = (title\_i or "").strip()

if len(label) &gt; 180:

label = label[:179] + "…"

cur.insert\_text((x, y), label, fontsize=FS, fontname=FONT, fill=COLOR)

pn = str(int(p1))

est = len(pn) \* FS \* 0.5

cur.insert\_text((cur.rect.width - RIGHT - est, y), pn, fontsize=FS, fontname=FONT, fill=(0,0,0))

clickable\_rect = fitz.Rect(x, y - FS, cur.rect.width - RIGHT, y + 4)

target\_idx = max(0, min((int(p1) - 1) + num\_toc\_pages, doc.page\_count - 1))

cur.insert\_link({

"kind": fitz.LINK\_GOTO, # internal “GoTo” link

"from": clickable\_rect, # clickable area

"page": target\_idx, # 0-based target page index

"zoom": zoom # fixed zoom (1.0=100%)

})

y += LINE\_H

try:

doc.set\_page\_labels([

{"startpage": 0, "prefix": "", "firstpagenum": 1, "style": "D"},

{"startpage": num\_toc\_pages, "prefix": "", "firstpagenum": 1, "style": "D"}

])

except Exception:

pass

out\_bytes = doc.tobytes()

finally:

doc.close()

return func.HttpResponse(

json.dumps({"status": "OK", "fileContent": base64.b64encode(out\_bytes).decode("ascii")}),

status\_code=200, mimetype="application/json")

except Exception as e:

logging.exception("TOC build failed")

return func.HttpResponse(json.dumps({"status": "ERROR", "error": str(e)}), status\_code=500, mimetype="application/json")

* PyMuPDF lets us create **internal GoTo links** via page.insert\_link({... 'kind': LINK\_GOTO ...}), specifying a clickable rectangle and target page—exactly what we use here. [[4]](https://pymupdf.qubitpi.org/en/latest/link.html)[[5]](https://github.com/pymupdf/PyMuPDF/discussions/1251)

**requirements.txt**

azure-functions

pymupdf==1.24.9

**host.json**

{

"version": "2.0"

}

**Example request and response format**

POST https://&lt;your-funcapp&gt;.azurewebsites.net/api/add-pdf-toc?code=&lt;function\_key&gt;

Content-Type: application/json

{

"fileContent": "&lt;BASE64\_of\_source\_pdf&gt;",

"title": "Hyperlinked Table of Contents",

"zoom": 1.0

}

{ "status": "OK", "fileContent": "&lt;BASE64\_of\_modified\_pdf&gt;" }

You can deploy with **VS Code** or **Azure Functions Core Tools**, then copy the **Function URL (with ?code=)** for use in your flow. The Python worker runs on a **Linux plan** in Azure. [[3]](https://learn.microsoft.com/en-us/azure/azure-functions/functions-reference-python)

**2) Power Automate (cloud) — Flow build sheet**

**Name:** Flow\_AddPdfToc\_Cloud\ **Trigger:** **Power Apps (V2)** with two **Text** inputs:

* sourcePath — OneDrive path like /Documents/pdf\_study/XL092002\_monthly\_tables\_pdf.pdf
* outputFolder — OneDrive folder like /Documents/pdf\_study

Power Apps (V2) lets you **strongly type inputs** (Text/Number/…); it’s the modern trigger to call flows from apps. [[1]](https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/how-to/trigger-flow)[[6]](https://4sysops.com/archives/using-power-automate-with-the-powerapps-v2-trigger/)

**Actions (in order):**

1. **OneDrive for Business – Get file content using path**
   * *File path* = **sourcePath** (from trigger)

The OneDrive for Business connector supports **Get file content using path** to fetch file bytes by a OneDrive-relative path. [[7]](https://manueltgomes.com/reference/power-automate-action-reference/get-file-content-using-path-action/)[[2]](https://learn.microsoft.com/en-us/connectors/onedriveforbusiness/)

1. **Compose** (Name: srcBase64)
   * **Inputs** = base64(body('Get*file*content*using*path')) (Converts the binary to Base64 for HTTP JSON.)
2. **HTTP** (to Azure Function)
   * **Method:** POST
   * **URI:** https://<your-funcapp>.azurewebsites.net/api/add-pdf-toc?code=<function\_key>
   * **Headers:** Content-Type: application/json
   * **Body:**
   * {
   * "fileContent": "@{outputs('srcBase64')}",
   * "title": "Hyperlinked Table of Contents",
   * "zoom": 1.0
   * }

Power Automate can call any HTTP endpoint (like your Function). You’ll POST Base64 and receive Base64 back. [[8]](https://azurelessons.com/call-azure-function-from-flow/)

1. **OneDrive for Business – Create file**
   * **Folder Path:** outputFolder (from trigger)
   * **File Name:** @{last(split(triggerBody()?['sourcePath'], '/'))} **replace** .pdf with \_with\_TOC.pdf (e.g., use a **Compose** before or inline)
   * **File Content:** base64ToBinary(body('HTTP')?['fileContent'])
2. **OneDrive for Business – Create share link for a file**
   * **File:** *(use the “Id” or “Path” dynamic value from Create file)*
   * **Link type:** View
   * **Scope:** Organization

The OneDrive for Business connector includes **Create share link** (view or edit; org scope). Perfect for returning a link to your app users. [[2]](https://learn.microsoft.com/en-us/connectors/onedriveforbusiness/)[[9]](https://www.matthewdevaney.com/how-to-create-a-sharing-link-for-files-in-power-automate/)

1. **Respond to a PowerApp or flow**
   * Add four **Text** outputs:
     + **status** = @{body('HTTP')?['status']}
     + **outputPath** = @{outputs('Create\_file')?['body/Path']}
     + **shareLink** = @{body('Create\_share\_link\_for\_a\_file')?['link']['webUrl']}
     + **error** = @{if(equals(body('HTTP')?['status'],'OK'),'', coalesce(body('HTTP')?['error'],'Unknown error'))}

Returning data to Power Apps via this action is the recommended pattern; then read properties off the returned object in your app. [[10]](https://www.microsoft.com/en-us/power-platform/blog/power-automate/return-data-to-powerapps/)

**Connector notes & limits:** The OneDrive for Business connector documents actions (get content, create file, create share link) and limitations (large files/timeouts), which may influence very large PDFs. [[2]](https://learn.microsoft.com/en-us/connectors/onedriveforbusiness/)

**3) Power Apps — Screen wiring (with Launch button)**

**Controls**

* txtSourcePath *(Text input; Default: /Documents/pdf\_study/XL092002\_monthly\_tables\_pdf.pdf)*
* txtOutputFolder *(Text input; Default: /Documents/pdf\_study)*
* btnCreateToc *(Button: “Create Hyperlinked TOC”)*
* btnOpenLink *(Button: “Open PDF”)*

**btnCreateToc.OnSelect**

Set(

varResult,

Flow\_AddPdfToc\_Cloud.Run(

{

sourcePath: txtSourcePath.Text,

outputFolder: txtOutputFolder.Text

}

)

);

If(

varResult.status = "OK",

Notify("TOC created", NotificationType.Success),

Notify("Error: " &amp; Coalesce(varResult.error, "Unknown error"), NotificationType.Error)

);

**btnOpenLink.DisplayMode**

If( IsBlankOrError(varResult.shareLink), Disabled, Edit )

**btnOpenLink.OnSelect**

Launch( varResult.shareLink )

The **Launch** function opens the **OneDrive sharing URL** in the user’s browser—exactly what you want for a “View PDF” experience. [[11]](https://learn.microsoft.com/en-us/power-platform/power-fx/reference/function-param)

**4) Test plan**

1. In **Power Automate**, run the flow **manually** once with test inputs:
   * sourcePath: /Documents/<your\_folder>/<file>.pdf
   * outputFolder: /Documents/<your\_folder>
2. Confirm the new file <file>\_with\_TOC.pdf appears in **OneDrive** and the first page is an inserted TOC with working links.
3. In **Power Apps**, click **Create Hyperlinked TOC**, then click **Open PDF** (Launch). It should open the **share link** in the browser.

**Notes & gotchas**

* **Bookmarks are required**: The TOC is created from **existing PDF bookmarks** (outline). If none exist, the function returns an error. (You can add bookmarks in your authoring workflow before running this.)
* **Org-only link**: The **Create share link** step is set to **Scope = Organization** and **Type = View** for easy internal sharing. You can change to other scopes or link types if needed. [[2]](https://learn.microsoft.com/en-us/connectors/onedriveforbusiness/)[[9]](https://www.matthewdevaney.com/how-to-create-a-sharing-link-for-files-in-power-automate/)
* **V2 trigger**: Use **Power Apps (V2)** for clean, typed inputs from your app. [[1]](https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/how-to/trigger-flow)
* **HTTP security**: Keep the Azure Function **authorization level** as **Function** and use the ?code= URL in the flow; store it in the flow or an environment variable/connection reference per your governance. [[3]](https://learn.microsoft.com/en-us/azure/azure-functions/functions-reference-python)

**Want me to hand you packaged assets?**

I can spin up a repo/zip with:

* The **Azure Function app** (files above: function\_app.py, requirements.txt, host.json, README.md)
* A **flow build guide** with screenshots
* A **Power Apps one‑pager** with formulas

Or, if you have an Azure resource group already set up, I can prepare a **Bicep/ARM** template or VS Code **launch.json/tasks.json** to let you press **Deploy**.

**Quick confirm:**

* Are you able to deploy an **Azure Function (HTTP)** in your tenant, or should I provide a ready‑to‑deploy zipped function + step‑by‑step VS Code instructions?
* Do you want the flow name fixed as Flow\_AddPdfToc\_Cloud and the app screen as scrPdfToc (matches your naming pattern)?

**References**

[1] [Create a canvas app that can trigger a Power Automate flow](https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/how-to/trigger-flow)

[2] [OneDrive for Business - Connectors | Microsoft Learn](https://learn.microsoft.com/en-us/connectors/onedriveforbusiness/)

[3] [Python developer reference for Azure Functions | Microsoft Learn](https://learn.microsoft.com/en-us/azure/azure-functions/functions-reference-python)

[4] [Link - PyMuPDF 1.26.3 documentation](https://pymupdf.qubitpi.org/en/latest/link.html)

[5] [Hyperlinks - Page.insert\_link - how to provide a zoomed ... - GitHub](https://github.com/pymupdf/PyMuPDF/discussions/1251)

[6] [Using Power Automate with the PowerApps V2 Trigger - 4sysops](https://4sysops.com/archives/using-power-automate-with-the-powerapps-v2-trigger/)

[7] [Power Automate: OneDrive for Business - Get File Content Using Path ...](https://manueltgomes.com/reference/power-automate-action-reference/get-file-content-using-path-action/)

[8] [Call Azure Function From Power Automate](https://azurelessons.com/call-azure-function-from-flow/)

[9] [How To Create A Sharing Link For Files In Power Automate](https://www.matthewdevaney.com/how-to-create-a-sharing-link-for-files-in-power-automate/)

[10] [Return data to PowerApps from a flow, build lists in a flow, and test a ...](https://www.microsoft.com/en-us/power-platform/blog/power-automate/return-data-to-powerapps/)

[11] [Launch and Param functions - Power Platform | Microsoft Learn](https://learn.microsoft.com/en-us/power-platform/power-fx/reference/function-param)