Swan – EIM Manual PDF Fetch (Repeatable Process)

This SOP (standard operating procedure) lets you repeatedly fetch GOV.UK Employment Income Manual (EIM) pages and save them as PDFs into the correct **parent folders**, with an `index.txt` per parent and a global `master_index.csv` for quick search.

---

0) What you'll get

- PDFs saved into: `C:\Users\Ben\swan-erp-system\swan-doc-vault\vault\guides\employment-income-general\<PARENT_FOLDER>\<CHILD_CODE>.pdf`
- Per-parent **index.txt** summarising contents (code, title, URL, filename)
- A global **master_index.csv** in the **base folder** for quick Excel filtering

Folder example:

```
employment-income-general
├─ EIM00500-Employment_income_contents
│  ├─ EIM00505.pdf
│  ├─ EIM01000.pdf
│  ├─ index.txt
└─ master_index.csv
```

---

1) One-time setup (VS Code + Python + Playwright)

1. **Install Python (3.10+)**
2. https://www.python.org/downloads/ → Install → tick **Add Python to PATH**.
3. **Install VS Code (if not installed)**
4. https://code.visualstudio.com/
5. **Open a working folder in VS Code**
6. e.g. `C:\Users\Ben\swan-erp-system\swan-doc-vault\vault\scripts` (create if needed)
7. **(Recommended) Create a virtual environment**
8. VS Code → `Ctrl+Shift+P` → **Python: Create Environment** → **Venv** → choose your Python 3.x.
9. Ensure the selected interpreter shows in the VS Code status bar.
10. **Install Playwright** (in VS Code Terminal)

    ```
    pip install playwright
    playwright install
    ```

    You only need to do steps 1–5 once per machine.

---

2) Base path & URL

- **Base path** (where files are written):

```
C:\Users\Ben\swan-erp-system\swan-doc-vault\vault\guides\employment-income-general
```

- **Manual base URL** (fixed):

```
https://www.gov.uk/hmrc-internal-manuals/employment-income-manual/
```

---

3) How to add/update EIM entries

You have **two ways** to manage the list of pages to fetch.

Option A (Simple): Edit list inside the script

- You'll paste new *parent* → *children* tuples into the `ENTRIES` section of the script.
- Use this when you only have a few changes and want to run immediately.

Option B (Scalable): Maintain a CSV next to the PDFs (recommended for big batches)

- Create a CSV called `master_entries.csv` in the **base path** with columns:

```
parent_code,parent_title,child_code,child_title
EIM00500,Employment income: contents,EIM00505,General: contents
EIM00500,Employment income: contents,EIM01000,Particular items: A to P: contents
EIM11300,Accommodation provided by reason of employment,EIM11300,Accommodation provided by reason of employment
EIM11300,Accommodation provided by reason of employment,EIM11342,Employer-provided living accommodation: detailed
```

- The script will **auto-detect** and use this CSV if present, ignoring the inline list.
- Keep adding rows as you upload screenshots and transcribe titles/codes.

Tip: you can build this CSV progressively. Each run refreshes the per-parent `index.txt` and the global `master_index.csv`.

---

4) The script (ready to run)

Save as `fetch_eim_manuals.py` in your scripts folder.

```
import os
import re
import csv
import time
from pathlib import Path
from datetime import datetime
from collections import defaultdict
from playwright.sync_api import sync_playwright, TimeoutError as
PlaywrightTimeoutError

# ===================== CONFIG =====================
BASE_URL = "https://www.gov.uk/hmrc-internal-manuals/employment-income-
manual/"
BASE_PATH = r"C:\\Users\\Ben\\swan-erp-system\\swan-doc-vault\\vault\\guides\
\employment-income-general"

# Fallback parent → children if CSV not present (Option A)
ENTRIES = [
    ("EIM00010", "Data protection", [
        ("EIM00010", "Data protection"),
    ]),
    ("EIM00100", "About this manual", [
        ("EIM00100", "About this manual"),
    ]),
    ("EIM00500", "Employment income: contents", [
        ("EIM00505", "General: contents"),
        ("EIM01000", "Particular items: A to P: contents"),
        ("EIM01650", "Particular items: New Deal, employment zones and
particular exemptions: contents"),
        ("EIM03050", "Employee Ownership Trusts – qualifying bonus payments:
introduction"),
        ("EIM03100", "Removal or transfer costs: contents"),
        ("EIM03600", "Restrictive covenants: contents"),
        ("EIM04700", "Particular items: R to Z: contents"),
    ]),
]
# ==================================================

INVALID = r'[<>:"/\\|?*\x00-\x1F]'
TRAILING = r'[\. ]+$'

def safe_title(title: str) -> str:
    t = title.replace(" ", "_").replace(":", "")
    t = re.sub(INVALID, "-", t)
    t = re.sub(TRAILING, "", t)
    return t

def write_parent_index(parent_folder: Path, parent_code: str, parent_title:
str, rows: list):
    lines = [
        f"{parent_code} — {parent_title}",
        f"Generated: {datetime.now().strftime('%Y-%m-%d %H:%M')}",
        "-" * 80
    ]
    for r in rows:
        lines.append(f"{r['child_code']} — {r['child_title']}")
        lines.append(f"  File : {r['filename']}")
        lines.append(f"  URL  : {r['url']}")
        lines.append("")
    (parent_folder / "index.txt").write_text("\n".join(lines),
encoding="utf-8")

def fetch_pdf(page, url: str, out_path: Path, tries: int = 3):
    for attempt in range(1, tries + 1):
        try:
            resp = page.goto(url, wait_until="networkidle", timeout=45_000)
            if not resp or resp.status >= 400:
                raise RuntimeError(f"HTTP {resp.status if resp else
'NoResponse'}")
            page.pdf(path=str(out_path), format="A4", print_background=True)
            return True
        except (PlaywrightTimeoutError, RuntimeError, Exception) as e:
            print(f"  Attempt {attempt}/{tries} failed: {e}")
            if attempt < tries:
                time.sleep(2 * attempt)
            else:
                return False
    return False

def load_entries_from_csv(base: Path):
    csv_path = base / "master_entries.csv"  # source of truth if present
    if not csv_path.exists():
        return None

    by_parent = defaultdict(lambda: {"title": None, "children": []})
    with csv_path.open("r", encoding="utf-8") as f:
        reader = csv.DictReader(f)
        for row in reader:
            pcode = row.get("parent_code", "").strip()
            ptitle = row.get("parent_title", "").strip()
            ccode = row.get("child_code", "").strip()
            ctitle = row.get("child_title", "").strip()
            if not pcode or not ptitle or not ccode:
                continue
            if by_parent[pcode]["title"] in (None, ""):
                by_parent[pcode]["title"] = ptitle
            by_parent[pcode]["children"].append((ccode, ctitle))

    entries = []
    for pcode, data in by_parent.items():
        entries.append((pcode, data["title"], data["children"]))
    return entries

def main():
    base = Path(BASE_PATH)
    base.mkdir(parents=True, exist_ok=True)

    # Use CSV (Option B) if present; otherwise fallback to inline ENTRIES
(Option A)
    loaded = load_entries_from_csv(base)
    entries = loaded if loaded else ENTRIES
    source_note = "CSV" if loaded else "inline list"
    print(f"Using entries from: {source_note}")

    master_csv = base / "master_index.csv"  # output index
    master_rows = []

    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        page = browser.new_page()

        for parent_code, parent_title, children in entries:
            parent_folder = base / f"{parent_code}-
{safe_title(parent_title)}"
            parent_folder.mkdir(exist_ok=True)

            parent_rows = []

            for child_code, child_title in children:
                url = BASE_URL + child_code.lower()
                pdf_path = parent_folder / f"{child_code}.pdf"

                print(f"Fetching {child_code} → {url}")
                ok = fetch_pdf(page, url, pdf_path)
                if ok:
                    print(f"  Saved: {pdf_path}")
                else:
                    print(f"  ❌ Failed: {child_code} ({url})")

                row = {
                    "parent_code": parent_code,
                    "parent_title": parent_title,
                    "child_code": child_code,
                    "child_title": child_title,
                    "filename": pdf_path.name,
                    "saved_path": str(pdf_path),
                    "url": url,
                }
                parent_rows.append({
                    "child_code": child_code,
                    "child_title": child_title,
                    "filename": pdf_path.name,
                    "url": url,
                    "saved_path": str(pdf_path),
                })
                master_rows.append(row)

            # write per-parent index files
            write_parent_index(parent_folder, parent_code, parent_title, 
parent_rows)

        browser.close()

    # write/update master index csv (output)
    with master_csv.open("w", newline="", encoding="utf-8") as f:
        writer = csv.DictWriter(
            f,

fieldnames=["parent_code","parent_title","child_code","child_title","filename","saved_path","u
        )
        writer.writeheader()
        writer.writerows(master_rows)

    print(f"\nMaster index written: {master_csv}")

if __name__ == "__main__":
    main()
```

---

5) Run workflow (every time)

1. **Open the workspace** in VS Code where `fetch_eim_manuals.py` lives.
2. **Activate venv** (if you created one) – VS Code usually auto-activates in the terminal.
3. **(Optional)** update `master_entries.csv` in the **base path** with new parent/child rows, or edit the inline list in the script.
4. **Run**:

```
python fetch_eim_manuals.py
```

5. **Review output**:
6. PDFs inside each **parent** folder
7. `index.txt` refreshed per parent
8. `master_index.csv` refreshed at the base path

---

6) How to build/maintain `master_entries.csv`

- Create (or open) `C:\Users\Ben\swan-erp-system\swan-doc-vault\vault\guides\employment-income-general\master_entries.csv`.
- Add/append rows:

```
parent_code,parent_title,child_code,child_title
EIM00500,Employment income: contents,EIM00505,General: contents
EIM00500,Employment income: contents,EIM01000,Particular items: A to P:
contents
EIM11300,Accommodation provided by reason of
employment,EIM11300,Accommodation provided by reason of employment
EIM11300,Accommodation provided by reason of
employment,EIM11342,Employer-provided living accommodation: detailed
```

- Save and re-run the script.

Rule of thumb: if you're doing more than ~10 additions at a time, use the CSV route.

---

7) Troubleshooting

- `ModuleNotFoundError: playwright`
- Run `pip install playwright` (inside your venv if using one) then `playwright install`.
- `TimeoutError` **or HTTP 4xx/5xx**
- GOV.UK may be slow or blocked. Re-run later. Script retries automatically with backoff.
- **Permissions / network path**
- Ensure you can create files in the base path.
- **Weird file/folder names**
- Script sanitises Windows-invalid characters; child **PDF** filenames use only the code (e.g. `EIM01000.pdf`).
- **Stop mid-run**: press `Ctrl + C` in the terminal.

---

8) Productivity add-ons (optional)

- **VS Code task**: create `.vscode/tasks.json` to one-click run.

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Fetch EIM PDFs",
      "type": "shell",
      "command": "python",
      "args": ["fetch_eim_manuals.py"],
      "problemMatcher": []
    }
  ]
}
```

- **Windows shortcut**: a `.bat` file to run the script.

```
@echo off
cd /d "C:\Users\Ben\swan-erp-system\swan-doc-
vault\vault\guides\employment-income-general"
python "C:\Users\Ben\swan-erp-system\swan-doc-
vault\vault\scripts\fetch_eim_manuals.py"
pause
```

---

9) Operating rhythm

1. Gather new pages (screenshots or codes).
2. Update `master_entries.csv` (preferred) **or** the inline `ENTRIES` list.
3. Run the script.
4. Use `master_index.csv` in Excel for quick search; jump straight to the parent folder.

That’s it. Repeat as needed without rework.