← .github/workflows/docs-intake.yml

Update docs-intake.yml #34

Workflow file

Workflow file for this run

.github/workflows/docs-intake.yml at 9f4c66e

```
#!/usr/bin/env python3
import sys, os, subprocess, hashlib, shutil
from pathlib import Path

def sha256(p: Path) -> str:
```

✕ Check failure on line 5 in .github/workflows/docs-intake.yml

GitHub Actions / .github/workflows/docs-intake.yml

Invalid workflow file

You have an error in your yaml syntax on line 5

```
    h = hashlib.sha256(); h.update(p.read_byt

def verify_manifest(folder: Path):
    mf = folder / "MANIFEST.sha256"
    if not mf.exists():
        print("[warn] No MANIFEST.sha256 in p
        return
    ok = True
    for raw in mf.read_text().splitlines():
        line = raw.strip()
        if not line: continue
        parts = line.split("  ", 1)  # "hash
        if len(parts) != 2:
            print("[ERR] Bad manifest line:",
        digest, name = parts
        fp = folder / name if not Path(name).
        if not fp.exists():
            print("[ERR] Missing file listed
        if sha256(fp) != digest:
            print("[ERR] Hash mismatch for",
    if not ok: sys.exit(2)
    print("[ok] Proposal manifest verified.")

def apply_patches(folder: Path) -> bool:
    """Return True only if ALL patches apply
    pd = folder / "patches"
    if not pd.exists(): return False
    patches = sorted(pd.glob("*.diff"))
    if not patches: return False

    # Pre-check each patch
    for p in patches:
        print(f"[info] Checking patch: {p.nam
        try:
            subprocess.run(["git", "apply", "
                           stdout=subprocess.
        except subprocess.CalledProcessError
            msg = e.stderr.decode(errors="ign
            print(f"[warn] Patch {p.name} not
            return False

    # All good → apply them
    for p in patches:
        print(f"[info] Applying patch: {p.nam
        subprocess.run(["git", "apply", "--in
    return True

def apply_proposals(folder: Path):
    for prop in folder.glob("*-proposal.md"):
        target = prop.name.replace("-proposal
        dst = (Path("docs") / target) if Path
        dst.parent.mkdir(parents=True, exist_
        shutil.copy2(prop, dst)
        print("[ok] Wrote", dst)
    for extra in ["release_notes.md", "CHANGE
        src = folder / extra
        if src.exists(): shutil.copy2(src, Pa

def main():
    if len(sys.argv) != 2:
        print("usage: apply_proposal.py <prop
    folder = Path(sys.argv[1]).resolve()
    if not folder.exists():
        print("proposal folder not found:", f

    verify_manifest(folder)

    used = False
    try:
        used = apply_patches(folder)
    except Exception as e:
        print(f"[warn] Patches threw exceptio

    if not used:
        print("[info] Using proposal files in
        apply_proposals(folder)

    # Rebuild MANIFEST for repo/docs
    root = Path("docs") if Path("docs").exist
    with open("MANIFEST.sha256","w",encoding=
        for p in sorted(root.glob("*.md")):
            mf.write(f"{sha256(p)}  {p}\n")

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