CONTRACT VALIDATOR COMMENTS

Absolutely—locking down **M5’s data contract** now is the right move, and we can also add a lightweight **M1 contract stub** so nothing upstream surprises us as we start M6.

Below I’m giving you:

1. **docs/m5\_contract.md** – the contract that matches the *actual, running* M5 (your current runner.py + engine.py).
2. **docs/m1\_contract.md (stub)** – minimal guardrails for M1 so we can expand it later without breaking M6–M9.
3. **Validator update** – a drop‑in update for tools/validate\_contracts.py to validate **M2 (inputs)** and **M5 (outputs)** together.
4. **Two one‑liners** – to run the validator and to spot‑check M5’s CFO reconciliation.

**1) docs/m5\_contract.md — Module 5 (Cash Flow) Artifact Contract – v1**

# Module 5 (M5) Artifact Contract – v1

\*\*Purpose.\*\* M5 computes \*\*Cash Flow from Operations (CFO)\*\* from M2 outputs and emits a monthly cash‑flow schedule used by M6 (Balance Sheet).

---

## Inputs (from M2, required)

Location: `./outputs/`

### A. P&L schedule (required)

- \*\*Canonical filename (preferred):\*\* `m2\_pl\_schedule.parquet`

- \*\*Accepted legacy name (tolerated):\*\* `m2\_profit\_and\_loss\_schedule.parquet`

\*\*Required roles (column synonyms)\*\*

M5 must be able to map these roles from the P&L file. Column names may be any of the synonyms below:

| Role | Synonyms accepted (case‑sensitive) | Units |

|--------------|-------------------------------------------------------------------------|-------|

| `MONTH\_INDEX`| `Month\_Index`, `MONTH\_INDEX` | index |

| `NPAT` | `NPAT\_NAD\_000`, `Net\_Profit\_After\_Tax\_NAD\_000`, `NPAT`, `Net\_Profit\_After\_Tax` | NAD’000 |

| `DA` | `Depreciation\_NAD\_000`, `Depreciation\_and\_Amortization\_NAD\_000`, `Depreciation`, `Depreciation\_and\_Amortization`, `DepreciationAmortization`, `DandA`, `DA` | NAD’000 |

### B. Working Capital schedule (required)

- \*\*Canonical filename (preferred):\*\* `m2\_working\_capital\_schedule.parquet`

- \*\*Accepted legacy name (tolerated):\*\* `m2\_working\_capital\_pl.parquet`

\*\*Required roles (column synonyms)\*\*

| Role | Synonyms accepted (case‑sensitive) | Units |

|--------------|-----------------------------------------------------------------------------------------------------|-------|

| `MONTH\_INDEX`| `Month\_Index`, `MONTH\_INDEX` | index |

| `NWC\_CF` | `Cash\_Flow\_from\_NWC\_Change\_NAD\_000`, `Net\_Working\_Capital\_CF\_NAD\_000`, `Working\_Capital\_CF\_NAD\_000`, `WC\_Cash\_Flow\_NAD\_000`, `Cash\_Flow\_from\_NWC\_Change`, `Net\_Working\_Capital\_CF`, `Working\_Capital\_CF`, `WC\_Cash\_Flow` | NAD’000 |

\*\*Typing & alignment\*\*

- `MONTH\_INDEX` is integer and \*\*must\*\* align across both files (same number of rows and month values).

- Monetary columns are numeric; \*\*units are thousands of NAD\*\* (NAD’000).

---

## Outputs (emitted by M5)

Location: `./outputs/`

1) \*\*Parquet:\*\* `m5\_cash\_flow\_statement\_final.parquet` (required)

\*\*Required columns\*\*

- `Month\_Index` (int)

- `NPAT\_NAD\_000` (float)

- `Depreciation\_NAD\_000` (float) ← mapped from `DA`

- `Cash\_Flow\_from\_NWC\_Change\_NAD\_000` (float) ← mapped from `NWC\_CF` (sign already consistent with cash flow)

- `CFO\_NAD\_000` (float) = `NPAT\_NAD\_000 + Depreciation\_NAD\_000 + Cash\_Flow\_from\_NWC\_Change\_NAD\_000`

\*\*Invariants\*\*

- No NaNs in required columns (M5 fills missing with zeros and logs it).

- `CFO\_NAD\_000` \*\*reconciles\*\* within 1e‑6 to the sum of components per month.

- Row count and `Month\_Index` equal to inputs’ `Month\_Index`.

2) \*\*Markdown:\*\* `m5\_smoke\_report.md` (required)

- Brief pass/fail notes and basic stats.

3) \*\*JSON (debug):\*\* `m5\_debug\_dump.json` (optional)

- Machine‑readable trace: resolved input paths, resolved role→column mapping, counts.

---

## Breaking‑change policy

- If a \*\*new synonym\*\* is introduced upstream (M2), \*\*runner must be updated\*\* to recognize it \*\*and\*\* this contract must be amended.

- \*\*Filenames\*\* are canonical above. Legacy names are tolerated \*\*for backward compatibility\*\*; new drops should adopt canonical names.

---

## Quick validation (human)

- Ensure `./outputs/m2\_pl\_schedule.parquet` & `./outputs/m2\_working\_capital\_schedule.parquet` exist.

- Ensure `./outputs/m5\_cash\_flow\_statement\_final.parquet` exists and includes the five required columns.

- Confirm `CFO\_NAD\_000` equals `NPAT + DA + NWC\_CF` for a few months spot‑checks.

**2) docs/m1\_contract.md — Module 1 Contract (stub)**

# Module 1 (M1) Artifact Contract – stub v0

\*\*Purpose.\*\* M1 computes operational schedules (yields, OPEX frames, etc.). \*\*As of M0–M5 stable\*\*, M1 has \*\*no direct downstream dependency\*\* enforced by contracts; M2 computes Working Capital and P&L independently from InputPack + drivers.

This stub exists to:

- Reserve canonical locations/filenames should M6–M9 later require M1 artifacts.

- Prevent accidental filename drift in future commits.

## (Reserved) Location and names

- Location (if/when emitted): `./outputs/`

- Suggested names (TBD if needed later):

- `m1\_operational\_summary.parquet`

- `m1\_smoke\_report.md`

## Policy

- Until a downstream module \*\*consumes\*\* an M1 artifact, \*\*no schema is frozen\*\*.

- When M6–M9 design first references any M1 column, we will promote this stub to a full contract (v1) with explicit role mappings, units, and invariants.

**3) Update tools/validate\_contracts.py to include M5 outputs**

Drop‑in replacement for the **new parts only** (keep your M2 validator that already passes). Append these functions and extend main() as shown.

# --- ADD near the top with the other imports ---

import pyarrow.parquet as pq

from typing import Dict, List, Tuple

# --- ADD helper used by both M2 and M5 checks (role resolution printouts etc.) ---

def \_assert\_columns(df\_cols: List[str], required: Dict[str, List[str]], label: str) -> Dict[str, str]:

"""

Ensure every role in 'required' is present in df\_cols under at least one synonym.

Returns role->resolved\_column mapping.

Raises AssertionError with a helpful message otherwise.

"""

cols\_set = set(df\_cols)

resolved = {}

missing = []

for role, syns in required.items():

hit = next((c for c in syns if c in cols\_set), None)

if hit:

resolved[role] = hit

else:

missing.append(role)

if missing:

pretty = json.dumps(required, indent=2)

raise AssertionError(

f"[{label}] Missing required roles for {label.split('/')[0]}: {missing}\n"

f"Looked for any of (normalized names): {pretty}\n"

f"Available columns: {df\_cols}"

)

return resolved

# --- ADD: M5 outputs validator ---

def validate\_m5\_outputs(out\_dir: str) -> Dict[str, Any]:

"""

Validates the M5 cash flow parquet and basic arithmetic identity.

"""

m5\_path = Path(out\_dir, "m5\_cash\_flow\_statement\_final.parquet")

if not m5\_path.exists():

raise FileNotFoundError(f"[M5] Expected output not found: {m5\_path}")

df = pq.read\_table(m5\_path).to\_pandas()

required\_cols = {

"MONTH\_INDEX": ["Month\_Index"],

"NPAT": ["NPAT\_NAD\_000"],

"DA": ["Depreciation\_NAD\_000"],

"NWC\_CF": ["Cash\_Flow\_from\_NWC\_Change\_NAD\_000"],

"CFO": ["CFO\_NAD\_000"],

}

mapping = \_assert\_columns(df.columns.tolist(), required\_cols, "M5/OUTPUT")

# Arithmetic identity: CFO = NPAT + DA + NWC\_CF

lhs = df[mapping["CFO"]]

rhs = df[mapping["NPAT"]] + df[mapping["DA"]] + df[mapping["NWC\_CF"]]

tol = 1e-6

if not ((lhs - rhs).abs() <= tol).all():

bad = (lhs - rhs).abs().max()

raise AssertionError(f"[M5] CFO reconciliation failed. max|diff|={bad}")

return {

"ok": True,

"file": m5\_path.name,

"rows": len(df),

"mapping": mapping,

}

# --- MODIFY main() to also run M5 ---

def main():

ap = argparse.ArgumentParser()

ap.add\_argument("out\_dir")

args = ap.parse\_args()

out\_dir = Path(args.out\_dir)

report = {}

# existing M2 validation you already have

report["m2"] = validate\_m2(out\_dir)

# new M5 validation

try:

report["m5"] = validate\_m5\_outputs(out\_dir)

except Exception as e:

report["m5\_error"] = f"{type(e).\_\_name\_\_}: {e}"

raise

dbg\_path = out\_dir / "contracts\_validate\_debug.json"

dbg\_path.write\_text(json.dumps(report, indent=2))

print(f" Wrote: {dbg\_path}")

print("🎉 All contracts passed.")

**4) One‑liners you can run now**

**A. Full contract validation (M2 + M5)**

.\.venv\Scripts\python.exe .\tools\validate\_contracts.py .\outputs

**B. Quick CFO identity spot‑check for 3 random rows**

.\.venv\Scripts\python.exe -c "import pyarrow.parquet as pq,random;import pandas as pd;d=pq.read\_table(r'.\outputs\m5\_cash\_flow\_statement\_final.parquet').to\_pandas();ix=random.sample(range(len(d)),min(3,len(d)));print(d.loc[ix,['Month\_Index','NPAT\_NAD\_000','Depreciation\_NAD\_000','Cash\_Flow\_from\_NWC\_Change\_NAD\_000','CFO\_NAD\_000']]);assert ((d['CFO\_NAD\_000']-(d['NPAT\_NAD\_000']+d['Depreciation\_NAD\_000']+d['Cash\_Flow\_from\_NWC\_Change\_NAD\_000'])).abs()<1e-6).all();print('OK: CFO identity holds')"

**Do we also need a contract for M1?**

* **Not required** for the currently working **M0–M5** pipeline and for the first pass of **M6** as we defined it (M6 consumes M2/M3/M4/M5).
* **Good practice** to have a stub so file names/locations don’t drift. That’s what docs/m1\_contract.md above provides.
* We’ll **promote** the M1 contract to a full schema **only when** a downstream module formally consumes an M1 column (we’ll then freeze names, units, and invariants).

**Why this matters for M6–M9**

* **M6 (Balance Sheet)** will trust the *frozen* M5 outputs (CFO\_NAD\_000 + components).
* With M5 contract in place, Coding Agent & Auditor can implement/check M6 logic without re‑opening M3/M5.
* Later, when you’re ready for **Capex & Equity wiring (Module “7.5”)**, we’ll add those roles/columns to the relevant module contracts and extend the validator to keep the identity checks airtight.

If you’d like, I can also produce the two contract files (docs/m5\_contract.md, docs/m1\_contract.md) and a ready-to-paste PR summary text so you can drop them into GitHub alongside the validator update.