# NFL25 Agent — Millions & Survivor Handoff (Full Docs v1.0)

This playbook is everything a new collaborator needs to pick up **Millions** (Contest Picks) alongside **Survivor** and the wider **NFL Agent** tooling. It consolidates architecture, weekly workflows, paste‑ready commands, known issues, and troubleshooting.

## 0) TL;DR Handoff

* **Single‑week workflow:** **A1 Build → A2 Enrich → A3 Kickoff helpers → Audit (safe baseline) → Export Member HTML.**
* **Sorting:** Cards sort by \_kickoff\_sort\_key (Thu→Fri→Sat→Sun→Mon last). Display text is \_kickoff\_pt.
* **DVOA scale:** Keep **fractions** in CSV (e.g., 0.095 = +9.5%). Exporter renders %.
* **Audit knobs:** --fail\_on\_missing 0.01, --spread\_range "-20,20", --dvoa\_range "-0.6,0.6", optional --roadmap + --line\_tolerance.
* **PowerShell quoting:** Use equals form for comma ranges: --spread\_range="-20,20" (prevents *expected one argument* errors).

## 1) Big Picture — Where Millions fits with Survivor & NFL Agent

**NFL Agent** is the umbrella; two member‑facing products sit under it:

* **Millions** (contest picks dashboard): weekly matchup cards showing **Favorite (Open/Current)**, **Circa Line** (TBD until contest posts), and **DVOA** (Total + Off + Def).
* **Survivor**: separate pipeline/logic for elimination picks; shares **schedule, aliases, and some audits** with Millions.

**Shared concepts/files:**

* **Team alias map** (e.g., WAS → WSH, ARZ → ARI, LA → LAR).
* **Clean schedule** (data/2025\_nfl\_schedule\_cleaned.csv) with week,date,time,vistm,hometm used to compute \_kickoff\_pt and \_kickoff\_sort\_key.
* **Audits**: structural sanity to keep the UI clean.

**Data flow alignment:** Roadmap (canon) → Planner (Millions working surface) → Exporter (HTML). Survivor has its own roadmap/expansion but can be cross‑checked in audits.

## 2) Repo Map (paths you’ll touch)

* picks/millions/millions\_roadmap\_game.csv — **canon schedule + lines** (season‑wide).
* picks/millions/millions\_planner.csv — **per‑team working surface** for exporter (single week filtered during build).
* picks/millions/exports/ — exported HTML files.
* picks/millions/diagnostics/ — audit outputs (\*.md, optional CSV flags).
* data/dvoa/2025\_dvoa\_projections.csv — DVOA source.
* data/2025\_nfl\_schedule\_cleaned.csv — season schedule (cleaned).
* scripts/ — Python scripts (build, audit, exporter, etc.).

## 3) Weekly Run — Pasteables (PowerShell)

Run all commands from repo root: C:\Users\Spencer\OneDrive\Desktop\nfl25-agent

### A1 — Build planner (single week)

python scripts/millions\_build\_planner.py ^  
 --season 2025 ^  
 --week 1 ^  
 --planner picks/millions/millions\_planner.csv ^  
 --schedule picks/millions/millions\_roadmap\_game.csv ^  
 --dvoa data/dvoa/2025\_dvoa\_projections.csv ^  
 --derive\_lines ^  
 --strip\_placeholders circa\_line result closing\_line line\_value pick\_side pick\_confidence notes

### A2 — Enrich planner (lines + Off/Def DVOA)

python - <<'PY'  
import pandas as pd  
from pathlib import Path  
  
planner\_p = Path(r"picks/millions/millions\_planner.csv")  
roadmap\_p = Path(r"picks/millions/millions\_roadmap\_game.csv")  
  
p = pd.read\_csv(planner\_p)  
r = pd.read\_csv(roadmap\_p)  
  
# Normalize team codes  
for df in (p,):  
 for c in ("team","opponent","home\_or\_away"):  
 if c in df.columns:  
 df[c] = df[c].astype(str).str.strip().str.upper()  
for c in ("home\_team","away\_team","hometm","vistm"):  
 if c in r.columns: r[c] = r[c].astype(str).str.strip().str.upper()  
  
# Orientation‑agnostic keys for joining  
p["\_key"] = p[["team","opponent"]].apply(lambda x: "::".join(sorted([x.team,x.opponent])), axis=1)  
if {"home\_team","away\_team"}.issubset(r.columns):  
 r["\_key"] = r[["home\_team","away\_team"]].apply(lambda x: "::".join(sorted([x.home\_team,x.away\_team])), axis=1)  
elif {"hometm","vistm"}.issubset(r.columns):  
 r["\_key"] = r[["hometm","vistm"]].apply(lambda x: "::".join(sorted([x.hometm,x.vistm])), axis=1)  
else:  
 raise SystemExit("Roadmap missing home/away team columns")  
  
# Columns to pull from roadmap (only those that exist)  
keep = [c for c in (  
 "kickoff\_local","venue",  
 "open\_spread\_home","open\_spread\_away",  
 "current\_spread\_home","current\_spread\_away",  
 "closing\_spread\_home","closing\_spread\_away",  
 "circa\_spread\_home","circa\_spread\_away",  
 "rest\_days\_home","rest\_days\_away","rest\_days\_diff",  
 "off\_dvoa\_home","off\_dvoa\_away","def\_dvoa\_home","def\_dvoa\_away"  
) if c in r.columns]  
  
meta = r[["\_key"] + keep].drop\_duplicates("\_key")  
  
# Drop overlaps to avoid \*\_x/\*\_y, merge  
p = p.drop(columns=[c for c in keep if c in p.columns], errors="ignore").merge(meta, on="\_key", how="left")  
  
# Derive team‑oriented Circa line (temporary)  
if 'home\_or\_away' in p.columns:  
 def pick\_circa(row):  
 if row.get('home\_or\_away') == 'HOME' and 'circa\_spread\_home' in p.columns:  
 return row.get('circa\_spread\_home')  
 if row.get('home\_or\_away') == 'AWAY' and 'circa\_spread\_away' in p.columns:  
 return row.get('circa\_spread\_away')  
 return row.get('circa\_line')  
 p['circa\_line'] = p.apply(pick\_circa, axis=1)  
  
# Map Off/Def DVOA into team/opponent perspective when available  
if {'off\_dvoa\_home','off\_dvoa\_away','def\_dvoa\_home','def\_dvoa\_away'}.issubset(p.columns):  
 def off\_map(row, side):  
 if row.get('home\_or\_away') == 'HOME':  
 return row.get('off\_dvoa\_home' if side=='team' else 'off\_dvoa\_away')  
 return row.get('off\_dvoa\_away' if side=='team' else 'off\_dvoa\_home')  
 def def\_map(row, side):  
 if row.get('home\_or\_away') == 'HOME':  
 return row.get('def\_dvoa\_home' if side=='team' else 'def\_dvoa\_away')  
 return row.get('def\_dvoa\_away' if side=='team' else 'def\_dvoa\_home')  
 p['team\_off\_dvoa\_proj'] = p.apply(lambda r: off\_map(r,'team'), axis=1)  
 p['opp\_off\_dvoa\_proj'] = p.apply(lambda r: off\_map(r,'opp'), axis=1)  
 p['team\_def\_dvoa\_proj'] = p.apply(lambda r: def\_map(r,'team'), axis=1)  
 p['opp\_def\_dvoa\_proj'] = p.apply(lambda r: def\_map(r,'opp'), axis=1)  
  
# Normalize Off/Def DVOA to fractions if they arrived in percent‑points  
for c in ("team\_off\_dvoa\_proj","team\_def\_dvoa\_proj","opp\_off\_dvoa\_proj","opp\_def\_dvoa\_proj",  
 "off\_dvoa\_home","off\_dvoa\_away","def\_dvoa\_home","def\_dvoa\_away"):  
 if c in p.columns:  
 s = pd.to\_numeric(p[c], errors="coerce")  
 if s.notna().any() and s.abs().max() > 1.5:  
 p[c] = s / 100.0  
  
p.drop(columns=['\_key'], inplace=True, errors='ignore')  
p.to\_csv(planner\_p, index=False)  
print("Planner enriched →", planner\_p)  
PY

### A3 — Kickoff helpers (PT text + robust sort key Thu→Mon)

python - <<'PY'  
import pandas as pd  
from pathlib import Path  
from datetime import datetime, timedelta  
  
PLANNER = Path(r"picks/millions/millions\_planner.csv")  
SCHED = Path(r"data/2025\_nfl\_schedule\_cleaned.csv")  
WEEK = 1  
  
p = pd.read\_csv(PLANNER)  
s = pd.read\_csv(SCHED)  
  
for c in ("team","opponent","home\_or\_away"):  
 if c in p.columns: p[c] = p[c].astype(str).str.strip().str.upper()  
  
s = s[s["week"]==WEEK].copy()  
for c in ("vistm","hometm"):  
 s[c] = s[c].astype(str).str.strip().str.upper().replace({"WAS":"WSH","ARZ":"ARI","LA":"LAR"})  
  
p["\_key"] = p[["team","opponent"]].apply(lambda r: "::".join(sorted([r.team,r.opponent])), axis=1)  
s["\_key"] = s[["vistm","hometm"]].apply(lambda r: "::".join(sorted([r.vistm,r.hometm])), axis=1)  
  
# Thu=0, Fri=1, Sat=2, Sun=3, Mon=4, Tue/Wed sink to 99  
map7 = {3:0,4:1,5:2,6:3,0:4,1:99,2:99}  
  
def make\_keys(row):  
 date\_raw = str(row.get("date")); time\_raw = str(row.get("time"))  
 if not date\_raw or not time\_raw:  
 return pd.Series({"kickoff\_pt":"", "kickoff\_sort\_key": float("inf")})  
 d=None  
 for dfmt in ("%m/%d/%Y","%Y-%m-%d"):  
 try:  
 d = datetime.strptime(date\_raw.split()[0], dfmt); break  
 except Exception: pass  
 if d is None: return pd.Series({"kickoff\_pt":"", "kickoff\_sort\_key": float("inf")})  
 t=None  
 for tfmt in ("%I:%M %p","%H:%M"):  
 try:  
 t = datetime.strptime(time\_raw.strip(), tfmt); break  
 except Exception: pass  
 if t is None: return pd.Series({"kickoff\_pt":"", "kickoff\_sort\_key": float("inf")})  
 et = datetime(d.year,d.month,d.day,t.hour,t.minute)  
 pt = et - timedelta(hours=3)  
 wd = map7.get(et.weekday(),99)  
 sort\_key = float(wd\*1440 + pt.hour\*60 + pt.minute)  
 disp = pt.strftime("%I:%M %p").lstrip("0") + " PT"  
 return pd.Series({"kickoff\_pt": disp, "kickoff\_sort\_key": sort\_key})  
  
meta = s.join(s.apply(make\_keys, axis=1))[['\_key','kickoff\_pt','kickoff\_sort\_key']].drop\_duplicates('\_key')  
for col in ["kickoff\_pt","kickoff\_sort\_key","\_kickoff\_pt","\_kickoff\_sort\_key"]:  
 if col in p.columns: del p[col]  
  
p = p.merge(meta, on="\_key", how="left")  
p["\_kickoff\_pt"] = p["kickoff\_pt"].replace({"": pd.NA})  
p["\_kickoff\_sort\_key"] = pd.to\_numeric(p["kickoff\_sort\_key"], errors="coerce")  
  
p.drop(columns=["\_key","kickoff\_pt","kickoff\_sort\_key"], errors="ignore", inplace=True)  
p.to\_csv(PLANNER, index=False)  
print("Kickoff helpers added →", PLANNER)  
PY

### B — Audit (safe baseline; optional lines check vs roadmap)

python -m scripts.millions\_audit\_week ^  
 --season 2025 ^  
 --week 1 ^  
 --planner "picks/millions/millions\_planner.csv" ^  
 --roadmap "picks/millions/millions\_roadmap\_game.csv" ^  
 --out\_dir "picks/millions/diagnostics" ^  
 --fail\_on\_missing 0.01 ^  
 --spread\_range "-20,20" ^  
 --dvoa\_range "-0.6,0.6" ^  
 --line\_tolerance 0.5

### C — Export Member HTML (sorted by kickoff; PT labels)

python scripts/millions\_export\_member\_html.py ^  
 --season 2025 ^  
 --week 1 ^  
 --planner "picks/millions/millions\_planner.csv" ^  
 --out "picks/millions/exports/member\_week\_1.html" ^  
 --sort\_by kickoff ^  
 --show\_rest 0

## 4) Exporter — what it expects & how it sorts

* **Home/Away labels** are derived from home\_or\_away, team, opponent.
* **Favorites** use \*\_spread\_home/away for open and current (negative = favorite). If missing, shows TBD.
* **Circa Line:** reads circa\_line (team‑oriented). If absent → TBD.
* **DVOA:** team\_total\_dvoa\_proj, opp\_total\_dvoa\_proj (and off/def if present). Exporter formats as %.
* **Kickoff:** uses planner \_kickoff\_pt and \_kickoff\_sort\_key **if present**. Only falls back to parsing kickoff\_local when those are missing.

If your HTML isn’t sorting correctly, ensure \_kickoff\_sort\_key exists in the planner and the exporter prefers it. (We patched attach\_display\_helpers to honor planner keys.)

## 5) Audit — what it checks today (safe baseline)

* **Schema:** season, week, team, opponent present.
* **Missingness:** For team, opponent, circa\_line, team\_total\_dvoa\_proj, opp\_total\_dvoa\_proj vs --fail\_on\_missing.
* **Ranges:** circa\_line in [-20,20]; DVOA totals in [-0.6,0.6].
* **HOME/AWAY** values sanity.
* **Duplicate matchups** across season,week,team,opponent.
* **Lines consistency vs roadmap** when --roadmap is passed (tolerance --line\_tolerance).

**Outputs**

* Markdown report: picks/millions/diagnostics/millions\_week\_audit.md.
* Optional CSV flags: picks/millions/diagnostics/millions\_week\_audit\_flags.csv.

## 6) Known Gotchas & One‑Shot Fixers (pasteables)

### A) DVOA shows 100×

Cause: Off/Def DVOA in **percent points**. Fix by normalizing to fractions:

python - <<'PY'  
import pandas as pd  
from pathlib import Path  
import shutil  
p\_path = Path(r"picks/millions/millions\_planner.csv")  
shutil.copy2(p\_path, p\_path.with\_suffix(".pre\_dvoa\_scale.bak.csv"))  
df = pd.read\_csv(p\_path)  
CANDS = [  
 "team\_off\_dvoa\_proj","team\_def\_dvoa\_proj",  
 "opp\_off\_dvoa\_proj","opp\_def\_dvoa\_proj",  
 "off\_dvoa\_home","off\_dvoa\_away","def\_dvoa\_home","def\_dvoa\_away",  
]  
changed = []  
for c in CANDS:  
 if c in df.columns:  
 s = pd.to\_numeric(df[c], errors="coerce")  
 if s.notna().any() and s.abs().max() > 1.5:  
 df[c] = s/100.0; changed.append(c)  
df.to\_csv(p\_path, index=False)  
print("Scaled columns:", changed)  
PY

### B) \*\_x/\*\_y duplicates after merges

python - <<'PY'  
import pandas as pd  
from pathlib import Path  
import shutil  
p\_path = Path(r"picks/millions/millions\_planner.csv")  
shutil.copy2(p\_path, p\_path.with\_suffix(".pre\_dedup.bak.csv"))  
df = pd.read\_csv(p\_path)  
suffix\_bases = sorted({c[:-2] for c in df.columns if c.endswith(("\_x","\_y"))})  
for base in suffix\_bases:  
 cx, cy = base+"\_x", base+"\_y"  
 if base not in df.columns: df[base] = pd.NA  
 if cy in df.columns: df[base] = df[base].where(df[base].notna(), df[cy])  
 if cx in df.columns: df[base] = df[base].where(df[base].notna(), df[cx])  
 for c in (cx, cy):  
 if c in df.columns: del df[c]  
# Drop truly duplicated names  
seen, keep\_cols = set(), []  
for c in df.columns:  
 if c not in seen: keep\_cols.append(c); seen.add(c)  
df = df.loc[:, keep\_cols]  
df.to\_csv(p\_path, index=False)  
print("Planner de-duplicated →", p\_path)  
PY

### C) PowerShell expected one argument for ranges

Use **equals** with quotes to keep the comma together:

--spread\_range "-20,20" --dvoa\_range "-0.6,0.6"

### D) Kickoff sorting still wrong in HTML

* Confirm planner has \_kickoff\_sort\_key and \_kickoff\_pt.
* Ensure exporter’s attach\_display\_helpers **prefers planner key** and only falls back to parsing.

### E) Schedule merge yields zero matches

* Confirm schedule is filtered to target week.
* Apply alias mapping (WAS→WSH, ARZ→ARI, LA→LAR).
* Join by orientation‑agnostic key built from sorted team pair.

## 7) PyCharm Usage & Copy/Paste Style

* **Terminal preference:** PowerShell with backticks (`) to continue lines. Backtick must be the **last char** on the line.
* **One‑liners:** Provide a one‑line variant whenever possible for quick reruns.
* **Inline Python blocks:** Use the here‑doc pattern so code is pasteable without local files:
* python - <<'PY'  
  # your python here  
  PY
* **Markdown preview:** Open .md outputs directly in PyCharm and enable preview in **Settings → Editor → Markdown**.
* **Run/Debug configs:** Add configs for A1 build, Audit, Exporter with working dir = repo root.
* **Formatting expectations:** Show **team codes** in dashboard; DVOA as **signed %**; missing fields should display **TBD** or em‑dash.

## 8) Current Status (Millions) & What’s Next

**Status:**

* Planner filtered to Week 1 and enriched with Off/Def DVOA and lines.
* \_kickoff\_pt + \_kickoff\_sort\_key present; exporter sorts by kickoff (Mon last).
* Audit green on required columns, ranges; Circa may be TBD until ingestion.
* Member HTML exports successfully.

**Up Next (Master Dashboard & polish):**

* Add **Totals (open/current/closing/circa\_total)** to roadmap → planner → exporter (display Current Total; optional delta from open).
* Tighten Master Dashboard audit flags: totals range check, line deltas, kickoff invariant, schema drift guardrail.
* Optional badges: injury/weather pills.

## 9) Survivor Tool — relation & hooks

* Survivor uses its own roadmap/expansion but shares **team aliases** and **clean schedule**.
* The Millions audit can optionally read survivor data for light presence checks.
* Integration points to keep aligned:
  + Team codes/aliases.
  + Week filtering and kickoff order.
  + Optional shared diagnostics folder for cross‑tool QA.

## 10) Field Reference (mini‑glossary)

* home\_or\_away: planner row perspective relative to team.
* \*\_spread\_home/away: game‑level lines (negative favors that side). Exporter computes **Favorite** from these.
* circa\_line: team‑oriented spread (negative = team favored). Derived from circa\_spread\_home/away until PDF ingestion.
* team\_\*\_dvoa\_proj / opp\_\*\_dvoa\_proj: **fractions**; exporter formats as %.
* \_kickoff\_pt: PT text for display; \_kickoff\_sort\_key: numeric weekday/time order (**Thu→Mon**).

## 11) QA Checklist before export

* Planner has **exactly** the week’s rows; no \*\_x/\*\_y columns.
* DVOA totals within [-0.6,0.6]; Off/Def look sane (rarely exceed ±0.3).
* Spreads in [-20,20].
* \_kickoff\_sort\_key numeric; sorting puts **Mon last**.
* Audit report shows **no red failures**.

## 12) Commit Guidance

* Commit runnable artifacts (scripts) and planner **after** audit passes.
* Keep diagnostics (picks/millions/diagnostics/\*) in repo or .gitignore per preference.
* Suggested message: > Week 1: build+enrich planner, kickoff keys, audit green, export member HTML

## 13) Appendix — Exporter patch (prefer planner kickoff key)

Ensure your attach\_display\_helpers in scripts/millions\_export\_member\_html.py **returns early** when \_kickoff\_sort\_key exists, and coerces it to numeric. Only fallback to parsing kickoff\_local if planner lacks the key. (This prevents HTML sort bugs.)

## 14) Appendix — Verify columns quickly

python - <<'PY'  
import pandas as pd  
p = pd.read\_csv(r"picks/millions/millions\_planner.csv")  
print("dup names:", p.columns[p.columns.duplicated()].tolist())  
print("has kickoff keys:", {c for c in ["\_kickoff\_pt","\_kickoff\_sort\_key"] if c in p.columns})  
for c in ('team\_off\_dvoa\_proj','team\_def\_dvoa\_proj','opp\_off\_dvoa\_proj','opp\_def\_dvoa\_proj'):  
 if c in p.columns:  
 s = pd.to\_numeric(p[c], errors='coerce')  
 if s.notna().any():  
 print(f"{c:22s} min={s.min(): .3f} max={s.max(): .3f} mean={s.mean(): .3f}")  
PY

**End of handoff.** This is the canonical reference for the current project state; keep it alongside the repo (e.g., docs/nfl25\_agent\_handoff\_v1.md).