Phase 1K

Got it. Here’s a tiny PR-style diff that implements the Phase 1K — Budget Trace (stub): a TS schema + factory, a minimal CLI that prints a single JSONL record (all zeros, offline), a deterministic test, and a script to run it.

```
*** Begin Patch
*** Update File: package.json
@@
   "scripts": {
+    "poc:budget": "tsx src/cli-budget.ts",
     "poc:diff": "node scripts/print-mock-diff.mjs",
     "test": "vitest run"
   },
+  "devDependencies": {
+    "tsx": "^4.16.0"
+  }
*** Add File: src/budgetTrace.ts
+export type BudgetStatus = 'stubbed' | 'computed' | 'final' | 'error';
+
+export interface BudgetTrace {
+  schemaVersion: '1.0.0';
+  provider: string;
+  model: string;
+  timestamp: string;        // ISO 8601 UTC
+  turnId: string;
+  runId: string;
+  inputTokens: number;
+  outputTokens: number;
+  totalTokens: number;
+  cacheMetrics: {
+    cacheCreationInputTokens: number;
+    cacheReadInputTokens: number;
+    cachedTokens: number;
+  };
+  costMicros: number;       // reserved for Phase 2
+  currency: string;         // e.g. "USD"
+  reasoningTokens: number;  // optional for o/claude-style models
+  tool: string;             // adapter identifier
+  status: BudgetStatus;
+}
+
+export function createStubBudgetTrace(
+  init?: Partial<BudgetTrace> & { turnId?: string; runId?: string }
+): BudgetTrace {
+  const now = new Date().toISOString();
+  const inputTokens = init?.inputTokens ?? 0;
+  const outputTokens = init?.outputTokens ?? 0;
+  const totalTokens = inputTokens + outputTokens;
+  return {
+    schemaVersion: '1.0.0',
+    provider: init?.provider ?? 'stub',
+    model: init?.model ?? 'unknown',
+    timestamp: init?.timestamp ?? now,
+    turnId: init?.turnId ?? 'turn_local_1',
+    runId: init?.runId ?? 'run_local_001',
+    inputTokens,
+    outputTokens,
+    totalTokens,
+    cacheMetrics: {
+      cacheCreationInputTokens:
+        init?.cacheMetrics?.cacheCreationInputTokens ?? 0,
+      cacheReadInputTokens: init?.cacheMetrics?.cacheReadInputTokens ?? 0,
+      cachedTokens: init?.cacheMetrics?.cachedTokens ?? 0,
+    },
+    costMicros: init?.costMicros ?? 0,
+    currency: init?.currency ?? 'USD',
+    reasoningTokens: init?.reasoningTokens ?? 0,
+    tool: init?.tool ?? 'stub_adapter',
+    status: init?.status ?? 'stubbed',
+  };
+}
```
*** *Add File: src/cli-budget.ts*
```
+#!/usr/bin/env node
+// Minimal, deterministic CLI for Phase 1K (offline stub)
+import { createStubBudgetTrace } from './budgetTrace.ts';
+
+// ultra-light arg parse: --turnId=foo --runId=bar
+const argPairs = Object.fromEntries(
+  process.argv.slice(2).map((a) => {
+    const [k, v] = a.split('=');
+    return [k.replace(/^--/, ''), v ?? ''];
+  }),
+);
+
+const bt = createStubBudgetTrace({
+  turnId: argPairs.turnId || 'turn_local_1',
+  runId: argPairs.runId || 'run_local_001',
+});
+
+// JSON Lines (one object per line) for CI/tooling
+process.stdout.write(JSON.stringify(bt) + '\n');
+
+// Stable status line for scripts to grep without JSON parsing
+process.stdout.write(
+  `BUDGETTRACE schemaVersion=${bt.schemaVersion} provider=${bt.provider} model=${bt.model}
status=${bt.status} totalTokens=${bt.totalTokens}\n`,
+);
```
*** *Add File: test/budgetTrace.test.ts*
```
+import { describe, it, expect, beforeEach, afterEach, vi } from 'vitest';
+import { createStubBudgetTrace } from '../src/budgetTrace';
+
+describe('budgetTrace stub', () => {
+  const fixed = new Date('2024-01-02T03:04:05.000Z');
+  beforeEach(() => vi.setSystemTime(fixed));
+  afterEach(() => vi.useRealTimers());
+
+  it('is deterministic with fixed clock and sums totals', () => {
+    vi.useFakeTimers();
+    const bt = createStubBudgetTrace({
+      turnId: 'turn_X',
+      runId: 'run_Y',
+      inputTokens: 12,
+      outputTokens: 8,
+    });
+    expect(bt.timestamp).toBe('2024-01-02T03:04:05.000Z');
+    expect(bt.totalTokens).toBe(20);
+    expect(bt.schemaVersion).toBe('1.0.0');
+    expect(bt.status).toBe('stubbed');
+    // stability of required fields
+    expect(bt).toMatchObject({
```

```
+      provider: 'stub',
+      model: 'unknown',
+      currency: 'USD',
+      tool: 'stub_adapter',
+      cacheMetrics: {
+        cacheCreationInputTokens: 0,
+        cacheReadInputTokens: 0,
+        cachedTokens: 0,
+      },
+    });
+  });
+});
+
*** End Patch
```

How to use

- Run the stubbed CLI (prints one JSONL record + a stable status line):

```
pnpm run poc:budget
# or with IDs
pnpm run poc:budget -- --turnId=turn_42 --runId=run_alpha
```

- Run the deterministic test:

```
pnpm test -t "budgetTrace stub"
```

This is Phase-1 compliant: offline, deterministic, CI-friendly JSON Lines, and matches your 1K schema (zeros acceptable).