Linear KPI Runner - Technical Implementation Notes

Table of Contents

System Overview

Linear KPI Runner is an intelligent CLI-based KPI tracking system for engineering teams using Linear project management. It provides:

- **DEL KPI**: Delivery Excellence Level metrics (Committed vs Completed deliverables)
- **Feature Movement**: Project state tracking (Done/In-Flight/Not Started)
- **Live Queries**: Real-time Linear API integration
- **Snapshot System**: Point-in-time data capture for historical analysis
- **Slack Integration**: Key discussions from team channels with user resolution
- **LLM Query Interpreter**: Natural language understanding with typo correction
- **Project Deep Dive**: Comprehensive project analysis combining Linear + Slack data
- **Historical Analysis**: Q1 retrospective data capture and trend tracking

Tech Stack

- **Runtime**: Node.js (ES6+)
- **Database**: SQLite (better-sqlite3)
- **API**: Linear GraphQL API, Slack Web API
- **LLM**: Fuelix API (GPT-5.2) for natural language queries
- **Cache**: File-based TTL cache + localStorage for dashboard

Supported Pods (9 Total)

Pod Name	Alias	Description
FTS	-	Full-Task Service
GTS	-	Ground Truth Service

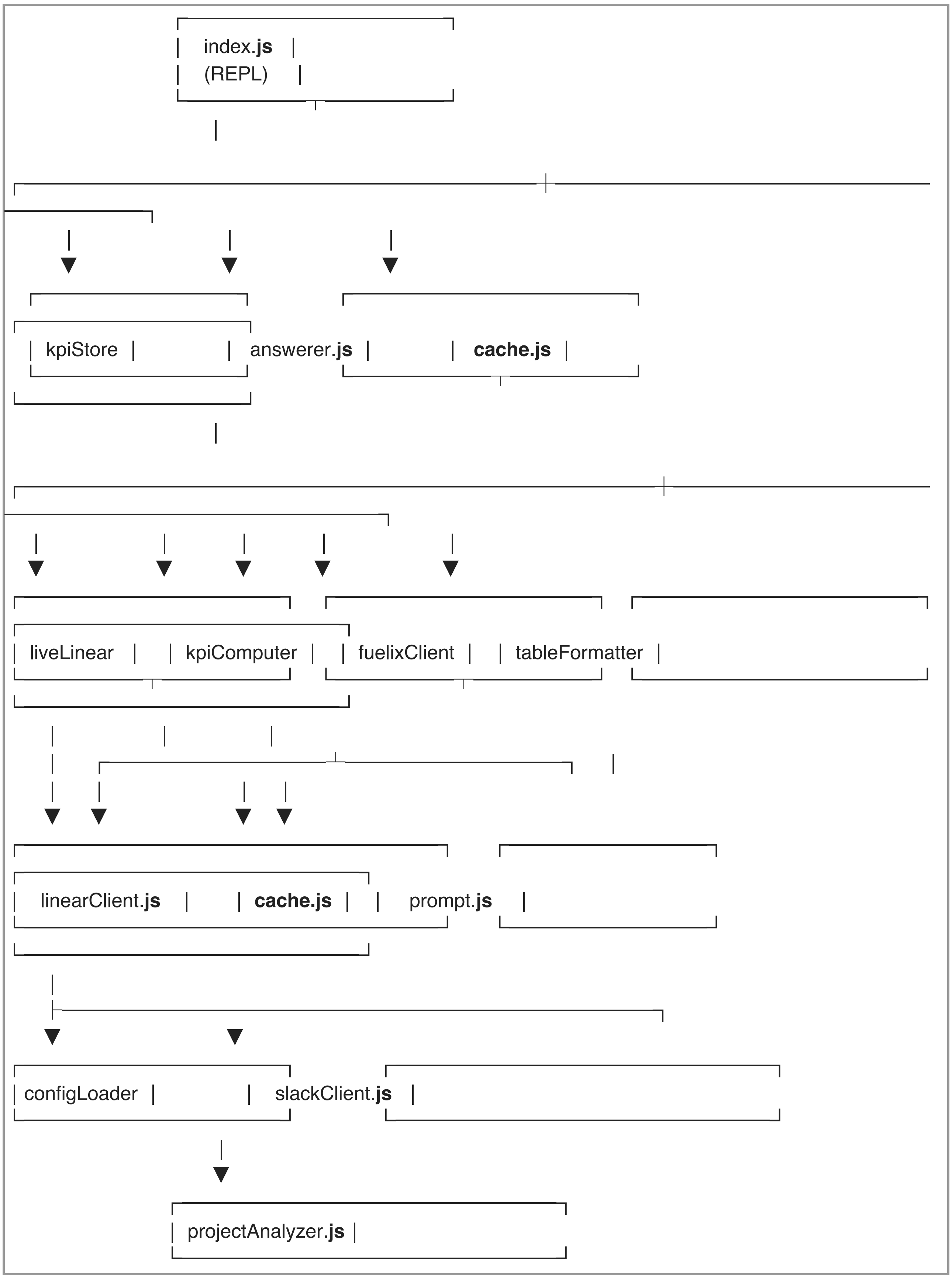
Platform	-	Platform Infrastructure
Control Center	-	Operations Control
Talent Studio	-	Talent Management
Growth & Reuse	GR	Growth & Reuse Team
ML	-	Machine Learning Team
FOT	-	Field Operations Technology
BTS	-	Business Technology Services

Architecture Diagram

High-Level System Architecture

```
┌──────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────┐
│                  LINEAR KPI RUNNER                            │
├──────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────┤
│                                              │
│  ┌──────────────────────────────┐    ┌──────────────────────────────┐
┌────────────────────────────────────────────────────────┐  │
│  │  CLI REPL   │     │  Scripts    │     │     External APIs       │  │
│  │  (index.js)  │     │            │     │                      │  │
│  └─────────────┬──────────────┘     │  ┌──────────────────────┐  │     │  ┌──────────────────┐
┌──────────────────────────┐  │  │
│      │         │  │ generate │  │     │  │ Linear │  │ Fuelix   │  │  │
│      │         │  │ Snapshot │  │     │ GraphQL │  │ LLM API  │  │  │
│      │         │  └──────────┬──────────┘  │   │  │ API │  │         │  │  │
│      │         │     │     │   │  └───────┬───────┘  └─────────┬─────────┘  │  │
│      │         │  ┌──────────┴──────────┐  │   │    │         │      │  │
│      │         │  │ runWeekly │  │    │  ┌───────┴───────┐      │      │  │
│      │         │  │   Kpi   │  │    │  │ Slack │      │      │  │
│      │         │  └──────────┬──────────┘  │   │  │ API │      │      │  │
│      │         └──────────┼──────────────┘     │  └────────┬────────┘      │      │  │
│      │            │          └──────────┼─────────────────────────┼───────────────┘  │
│      │            │            │         │       │
│
┌─────────────▼─────────────────────────────────────▼─────────────────────────────
────▼──────────────────────────▼──────────────────┐  │
│  │             CORE ENGINE                       │  │
│  │  ┌──────────────────────────┐  ┌──────────────────────────┐
┌──────────────────────────┐  ┌──────────────────────────┐  │  │
│  │  │  Answerer  │  │ KPI Computer │  │ Live Linear │  │  Slack   │  │  │
│  │  │  Engine   │◄──┤   Engine   │◄──┤  Fetcher   │◄──┤  Client   │  │  │
│  │  └──────────────────────────┘  └──────────────────────────┘
└─────────────┬─────────────┘  └──────────────────────────┘  │  │
│  │      │                  │               │  │
│  │      │         ┌───────────────────────────────────┴───────────────────────────────┐
│  │
│  │      │         │                   │       │  │
│   │   ┌─────────────▼──────────┐   ┌─────────▼──────────┐   ┌────────────────────────────┐
┌────────────────▼───────────────┐   │   │
│   │   │  Project   │   │  Linear  │   │  Config   │   │  Fuelix   │   │   │
│   │   │ Analyzer   │   │  Client  │   │  Loader   │   │  Client   │   │   │
│   │   └────────────────────────┘   └──────┬──────────┘   └──────────┬─────────────┘
└───────────────────────────────┘   │   │
│
└─────────────────────────────────────┼────────────────────────────┼──────
──────────────────────────────────────┘   │
│                │          │                   │
│
┌─────────────────────────────────────▼────────────────────────────▼──────
──────────────────────────────────────┐   │
│   │           DATA LAYER                        │   │
│   │   ┌──────────────────────────┐   ┌──────────────────────────┐
┌──────────────────────────┐   ┌──────────────────────────┐   │   │
│   │   │  SQLite   │   │  Snapshot  │   │  Config   │   │  Cache   │   │   │   │
│   │   │ Database  │   │    JSON    │   │   Files   │   │  Files   │   │   │   │
│   │   │ (state/db) │   │  (output/)  │   │ (config/) │   │ (cache/)  │   │   │   │
│   │   └──────────────────────────┘   └──────────────────────────┘
└──────────────────────────┘   └──────────────────────────┘   │   │
│
└──────────────────────────────────────────────────────────────────────
──────────────────────────────────────┘   │
│                                      │
└──────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────┘
```

Module Dependency Graph



Data Flow Architecture

```
│                    USER INPUT FLOW                          │
└──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────┘

User Input         LLM Interpreter      Answer            Output
     │                    │              Engine              │
     │                    │                 │                │
     ▼                    ▼                 ▼                ▼
┌───────────────┐     ┌────────────────────────┐    ┌──────────────────┐
┌──────────────────┐
│  "kpi"    │─────────▶ │ interpretQuery-  │──▶ │  answer()    │───────▶ │  Formatted   │
│  "pod X"  │        │   WithLLM()      │    │         │     │  Output     │
│  "proj Y" │        │              │   │  switch on  │      │          │
└───────────────┘     │  Typo correction │    │  cmd.type    │    │  Markdown    │
      │  Intent extract  │   │        │    │  Tables      │
      └────────────────────────┘  └──────────┬──────────┘
└──────────────────┘
                          │
      ┌───────────────────────────────────────────┼──────────────────────────────┐
      │                    │              │
      ▼                    ▼              ▼
    ┌──────────────────┐      ┌──────────────────┐
┌──────────────────┐
      │  Snapshot    │      │  Live API    │     │  Project    │
      │  Lookup      │      │  Fetch       │     │  Deep Dive  │
      └──────────────────┘      └──────────────────┘
└────────┬─────────┘
                                │
              ┌─────────────────────────────────┴──────────────────────────┐
              │                       │
              ▼                       ▼
          ┌──────────────────┐      ┌──────────────────┐
          │   Slack     │             │  Linear    │
          │  Messages   │              │  Comments   │
          └──────────────────┘      └──────────────────┘


┌──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────┐
│                    KPI COMPUTATION FLOW                          │
└──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────┘

┌───────────────┐     ┌────────────────────┐     ┌──────────────────┐
┌──────────────────────┐
│  Config   │─────────▶ │  Load Pods &  │─────────▶ │  Fetch DEL     │─────────▶ │  Compute      │
│  Files    │     │  Label IDs   │     │  Issues     │     │  Metrics      │
└───────────────┘     └────────────────────┘     └──────────────────┘
└──────────┬───────────┘
                                  │
┌──────────────────────────────────────────────────────────────────
```

```
└──────────────────┘
   │
   ▼
┌───────────────────────┐   ┌───────────────────────┐
┌───────────────────────┐
│  For Each    │───────▶│  Count       │───────▶│  Calculate    │
│  Cycle C1-C6  │     │ Committed/   │     │ Delivery %   │
│          │     │ Completed    │     │ & Spillover  │
└───────────────────────┘   └───────────────────────┘
└───────────────────────┘
                 │
                 ▼
                ┌───────────────────────┐
                │  Format Table  │
                │  + Insights    │
                └───────────────────────┘
```

Component Architecture

1. CLI Layer (agent/src/index.js)

```
┌─────────────────────────────────────────────────────────────────┐
│            index.js                  │
├─────────────────────────────────────────────────────────────────┤
│  - REPL interface (readline)            │
│  - Command routing                      │
│  - Slash commands (/help, /refresh)     │
│  - Answer logging                       │
├─────────────────────────────────────────────────────────────────┤
│  Dependencies:                          │
│   ├── answerer.js                       │
│   ├── kpiStore.js                       │
│   ├── cache.js                          │
│   └── liveLinear.js                     │
└─────────────────────────────────────────────────────────────────┘
```

2. Answer Engine (agent/src/answerer.js)

```
┌─────────────────────────────────────────────────────────────────┐
│          answerer.js                    │
├─────────────────────────────────────────────────────────────────┤
│  COMMAND PARSER                          │
│   ┌───────────────────────────────────────────────────────┐   │
│   │ parseCommand(input)               │    │
│   │   → combined_kpi | weekly_kpi      │    │
│   │   → all_pods_summary (cross-pod)    │    │
│   │   → pod_summary | pod_projects      │    │
│   │   → pod_narrative (rich output)     │    │
│   │   → project_detail | blockers       │    │
│   │   → project_deep_dive (Slack+Lin)   │    │
│    │   → project_comments | dels_by_cyc  │    │
│    │   → pending_dels | del_kpi         │    │
│    │   → debug_mode | clear_cache        │    │
│    └────────────────────────────────────────────────────────┘    │
├──────────────────────────────────────────────────────────────────┤
│  LLM QUERY INTERPRETER                     │
│    ┌────────────────────────────────────────────────────────┐    │
│    │  interpretQueryWithLLM()            │    │
│    │   - Typo correction                  │    │
│    │   - Intent extraction               │    │
│    │   - Entity recognition               │    │
│    │   - Confidence scoring               │    │
│    └────────────────────────────────────────────────────────┘    │
├──────────────────────────────────────────────────────────────────┤
│  POD/PROJECT DETECTION                     │
│    ┌────────────────────────────────────────────────────────┐    │
│    │  isPodName() - Exact pod match      │    │
│    │  extractPodFromQuery() - Find pod   │    │
│    │  extractProjectFromNaturalLang()    │    │
│    │  Supports: FTS, GTS, Platform,      │    │
│    │    Control Center, Talent Studio,   │    │
│    │    Growth & Reuse, ML, FOT, BTS     │    │
│    └────────────────────────────────────────────────────────┘    │
├──────────────────────────────────────────────────────────────────┤
│  ANSWER STRATEGIES                         │
│    ┌────────────────────────────────────────────────────────┐    │
│    │  1. Deterministic (snapshot-based)  │    │
│    │  2. Live API fetch                  │    │
│    │  3. Project Deep Dive (Slack+Lin)   │    │
│    │  4. LLM fallback                    │    │
│    └────────────────────────────────────────────────────────┘    │
├──────────────────────────────────────────────────────────────────┤
│  FEATURE READINESS TRACKING                │
│    ┌────────────────────────────────────────────────────────┐    │
│    │  fetchPodFeatureReadiness()         │    │
│    │   - PRD status (done/in_progress)   │    │
│    │   - Design status                   │    │
│    │   - BE/FE Dev status                │    │
│    │   - PAT/QA status                   │    │
│    │   - Tech Debt categorization        │    │
│    └────────────────────────────────────────────────────────┘    │
├──────────────────────────────────────────────────────────────────┤
│  FORMATTERS (uses tableFormatter.js)      │
│    ├── formatPodSummary()                 │
│    ├── formatProjectList() → box tables   │
│    ├── formatProjectDetail()              │
│    ├── formatBlockers() → box tables      │
│    ├── formatPodsList() → box tables      │
│    └── generateAllPodsSummary()           │
└──────────────────────────────────────────────────────────────────┘
```

3. KPI Computer (agent/src/kpiComputer.js)

```
┌──────────────────────────────────────────────────────────────┐
│          kpiComputer.js                    │
├──────────────────────────────────────────────────────────────┤
│  COMPUTATION FUNCTIONS                     │
│   ┌────────────────────────────────────────────────────┐   │
│   │ computeCycleKpi()                 │    │
│   │   └── DEL metrics per pod/cycle      │    │
│   │                                │    │
│   │ computeFeatureMovement()           │    │
│   │   └── Project states per pod        │    │
│   │                                │    │
│   │ computeCombinedKpi()              │    │
│   │   └── Both + project summaries      │    │
│   │                                │    │
│   │ fetchPendingDELs()                │    │
│   │   └── Get incomplete DELs by pod    │    │
│   │                                │    │
│   │ fetchDELsByCycle()                │    │
│   │   └── DELs planned for C1-C6        │    │
│   └────────────────────────────────────────────────────┘   │
├──────────────────────────────────────────────────────────────┤
│  FORMATTERS                          │
│   ├── formatCycleKpiTable()              │
│   ├── formatFeatureMovementTable()         │
│   ├── formatCombinedKpiOutput()            │
│   ├── formatPendingDELs()                │
│   ├── formatDELsByCycle()                │
│   └── generateInsights()               │
├──────────────────────────────────────────────────────────────┤
│  HELPERS                             │
│   ├── getCycleKeyByDate()                │
│   ├── isCycleActive()                  │
│   └── getBestCycleByCommitted()           │
└──────────────────────────────────────────────────────────────┘
```

4. Linear Client (agent/src/linearClient.js)

```
┌──────────────────────────────────────────────────────────────┐
│          linearClient.js                    │
├──────────────────────────────────────────────────────────────┤
│  class LinearClient                        │
│   ┌──────────────────────────────────────────────────────┐   │
│   │ constructor({ apiKey, url })      │   │
│   │                                   │   │
│   │ gql(query, variables)             │   │
│   │   └── Execute GraphQL query       │   │
│   │                                   │   │
│   │ findTeamByName(name)              │   │
│   │ getProjectsByInitiative(id)       │   │
│   │ getIssuesByTeam(teamId)           │   │
│   │ getIssuesByProject(projectId)     │   │
│   │ getProjectById(projectId)         │   │
│   │ getIssueComments(issueId, limit)  │   │
│   │ searchProjects(query, limit)      │   │
│   │ getFeatureReadiness(projectId)    │   │
│   └──────────────────────────────────────────────────────┘   │
└──────────────────────────────────────────────────────────────┘
```

5. Slack Client (agent/src/slackClient.js)

```
┌──────────────────────────────────────────────────────────┐
│        slackClient.js              │
├──────────────────────────────────────────────────────────┤
│  class SlackClient                   │
│  ┌────────────────────────────────────────────────┐  │
│  │ constructor({ botToken })       │  │
│  │                       │  │
│  │ api(method, params, options)      │  │
│  │   └── Supports JSON & form-encoded   │  │
│  │                       │  │
│  │ testAuth()                 │  │
│  │ joinChannel(channelId)          │  │
│  │ getChannelInfo(channelId)        │  │
│  │                       │  │
│  │ getMessages(channelId, options)    │  │
│  │ getAllMessages(channelId, opts)    │  │
│  │   └── With pagination (unlimited)    │  │
│  │                       │  │
│  │ getThreadReplies(channelId, ts)    │  │
│  │ getAllThreadReplies(channelId,ts)  │  │
│  │ getMessagesWithThreads()         │  │
│  │   └── Expands all thread replies     │  │
│  │                       │  │
│  │ getUserInfo(userId)            │  │
│  │ resolveUserNames(userIds)        │  │
│  │   └── Batch user ID → name mapping   │  │
│  │                       │  │
│  │ postMessage(channelId, text)      │  │
│  │ postRichMessage(channelId, block)  │  │
│  └────────────────────────────────────────────────┘  │
├──────────────────────────────────────────────────────────┤
│  STATIC HELPERS                    │
│  ├── formatTimestamp(ts)              │
│  └── extractUserIds(messages)           │
└──────────────────────────────────────────────────────────┘
```

6. Project Analyzer (agent/src/projectAnalyzer.js)

```
┌──────────────────────────────────────────────────────────┐
│        projectAnalyzer.js                │
├──────────────────────────────────────────────────────────┤
│  class ProjectAnalyzer                   │
│  ┌─────────────────────────────────────────────┐   │
│  │ Combines Slack + Linear data into │   │
│  │ unified project analysis          │   │
│  └─────────────────────────────────────────────┘   │
├──────────────────────────────────────────────────────────┤
│  analyzeProject(projectId, options)      │
│  ├── daysBack: default 14 days           │
│  ├── includeThreads: true/false          │
│  │                               │
│  │ Returns:                         │
│  │  ├── project info                 │
│  │  ├── slack: messageCount, threadCount  │
│  │  │       participants, totalReplies  │
│  │  ├── linear: issueCount, commentCount  │
│  │  │        participants              │
│  │  └── timeline: unified chronological   │
│  │           events from both sources │
├──────────────────────────────────────────────────────────┤
│  INTERNAL METHODS                        │
│  ├── _fetchSlackData(channelId, opts)     │
│  ├── _fetchLinearData(projectId)         │
│  └── _buildTimeline(slack, linear)       │
└──────────────────────────────────────────────────────────┘
```

7. Live Linear Layer (agent/src/liveLinear.js)

```
┌─────────────────────────────────────────────────────────────┐
│         liveLinear.js                   │
├─────────────────────────────────────────────────────────────┤
│  SINGLETON MANAGEMENT                      │
│   ├── getClient()  → LinearClient          │
│   └── getConfig()  → Config object         │
├─────────────────────────────────────────────────────────────┤
│  LIVE DATA FUNCTIONS (cached)              │
│   ┌─────────────────────────────────────────────────────┐   │
│   │ getLiveProjects(podName)          │   │
│   │ getLiveProject(pod, projectQuery)  │   │
│   │ getLiveBlockers(pod, project)      │   │
│   │ getLiveComments(pod, proj, days)   │   │
│   │ getLivePodSummary(podName)         │   │
│   │ listPods()                         │   │
│   └─────────────────────────────────────────────────────┘   │
├─────────────────────────────────────────────────────────────┤
│  UTILITIES                              │
│   ├── normalizeState()                  │
│   ├── classifyIssue()                   │
│   ├── scoreProjectMatch()               │
│   ├── fuzzyMatchProject()               │
│   ├── cacheStats()                      │
│   └── clearCache()                      │
└─────────────────────────────────────────────────────────────┘
```

KPI Computation Logic

DEL (Delivery Excellence Level) Metrics

```
┌──────────────────────────────────────────────────────────────────────
─────────────────────────┐
│                DEL COMPUTATION                    │
├──────────────────────────────────────────────────────────────────────
─────────────────────────┤
│                                     │
│  COMMITTED = Issues with:                         │
│    ├── DEL label (e.g., "DEL")                    │
│    ├── Cycle baseline label (e.g., "2026Q1-C1")          │
│    └── NOT DEL-CANCELLED label                       │
│                                     │
│  COMPLETED = Committed issues where:                     │
│    ├── state.type === "completed"                  │
│    └── completedAt <= cycleEnd (if cycle closed)          │
│       OR completedAt <= now (if cycle active)          │
│                                     │
│  SPILLOVER = (cycle active) ? 0 : committed - completed  │
│                                     │
│  DELIVERY % = (completed / committed) * 100              │
│                                     │
└──────────────────────────────────────────────────────────────────────
─────────────────────────┘
```

Cycle Detection Algorithm

```
function getCycleKeyByDate(podCalendar, refDate = new Date()) {
  // 1. Find active cycle (date within [start, end])
  for (let i = 1; i <= 6; i++) {
    const c = podCalendar[`C${i}`];
    if (refDate >= c.start && refDate <= c.end) {
      return `C${i}`;  // Active cycle found
    }
  }

  // 2. No active cycle - find most recently ended
  let best = null, bestEnd = -Infinity;
  for (let i = 1; i <= 6; i++) {
    const c = podCalendar[`C${i}`];
    if (c.end <= refDate && c.end > bestEnd) {
      bestEnd = c.end;
      best = `C${i}`;
    }
  }

  return best || "C1";  // Default fallback
}
```

Feature Movement Computation

```
┌─────────────────────────────────────────────────────────────────
──────────────────────────┐
│            FEATURE MOVEMENT STATES                    │
├─────────────────────────────────────────────────────────────────
─────────────────────────┤
│                                                    │
│   Project State          │   Normalized State              │
│
──────────────────────────────────────────────────┼──────────────────────────────────
──────────────── │
│   "completed"            │   "done"                          │
│   "started", "paused"    │   "in_flight"                     │
│   "planned", "backlog"   │   "not_started"                   │
│   "canceled"             │   "cancelled"                     │
│                                                    │
│   Per Pod:                                              │
│     plannedFeatures = total projects                    │
│     done = count(state == "done")                       │
│     inFlight = count(state == "in_flight")              │
│     notStarted = count(state == "not_started")          │
│                                                    │
└─────────────────────────────────────────────────────────────────
──────────────────────────┘
```

Table Formatting System

Overview

The tableFormatter.js module provides beautiful Unicode box-drawing tables for CLI output, replacing plain markdown tables with visually appealing, aligned tables.

Architecture

```
┌──────────────────────────────────────────────────────────────┐
│       tableFormatter.js              │
├──────────────────────────────────────────────────────────────┤
│  BOX DRAWING CHARACTERS                     │
│  ┌────────────────────────────────────────────────────┐   │
│  │  ┌┐  └┘  ─  │  ├┤  ┬┴┼          │   │
│  │  topLeft, topRight, bottomLeft...  │   │
│  └────────────────────────────────────────────────────┘   │
├──────────────────────────────────────────────────────────────┤
│  CORE FORMATTER                            │
│  ┌────────────────────────────────────────────────────┐   │
│  │  formatTable(data, columns, opts)   │   │
│  │   - Auto column width calculation   │   │
│  │   - Left/Right/Center alignment     │   │
│  │   - Optional title & totals row     │   │
│  └────────────────────────────────────────────────────┘   │
├──────────────────────────────────────────────────────────────┤
│  SPECIALIZED FORMATTERS                    │
│  ├── formatFeatureMovementBox()          │
│  │    └── Pod feature state table          │
│  ├── formatDelKpiBox()                   │
│  │    └── DEL KPI metrics table            │
│  ├── formatPendingDelsBox()              │
│  │    └── Pending DEL issues table         │
│  ├── formatProjectsBox()                 │
│  │    └── Project list table               │
│  ├── formatBlockersBox()                 │
│  │    └── Blocker issues table             │
│  ├── formatPodsListBox()                 │
│  │    └── Available pods table             │
│  └── formatSummaryBox()                  │
│       └── Simple summary box               │
├──────────────────────────────────────────────────────────────┤
│  UTILITIES                                 │
│  ├── padCell(value, width, align)        │
│  └── truncate(str, maxLen)               │
└──────────────────────────────────────────────────────────────┘
```

Example Output

Feature Movement (Weekly Snapshot)

Pod	Planned	Done	In-Flight	Not Started
FTS	17	2	4	11
GTS	8	1	3	4
Platform	13	3	2	8
Control Center	11	1	6	4
Talent Studio	12	2	4	6
Growth & Reuse	10	0	1	9
ML	0	0	0	0
FOT	6	0	1	5
BTS	0	0	0	0
TOTAL	77	9	21	47

Slack Integration

Overview

The Slack integration enables fetching team discussions, thread replies, and resolving user IDs to names for better readability in reports.

Key Features

1. **Message Fetching**: Retrieve channel messages with pagination
2. **Thread Expansion**: Get all replies for threaded conversations
3. **User Resolution**: Convert <@U12345> mentions to actual names
4. **Rate Limiting**: Built-in delays to respect Slack API limits

User ID Resolution Flow

```
┌──────────────────────────────────────────────────────────────
───────────────────────────┐
│           USER ID RESOLUTION FLOW                    │
├──────────────────────────────────────────────────────────────
───────────────────────────┤
│                                                     │
│  Input: "Thanks <@U1234> for the update on <@U5678>'s    │
│         ticket"                                       │
│                                                     │
│  Step 1: Extract User IDs                              │
│     └── Found: [U1234, U5678]                         │
│                                                     │
│  Step 2: Batch Resolve via Slack API                   │
│     └── users.info (form-urlencoded) for each ID         │
│     └── Results: { U1234: "John", U5678: "Sarah" }         │
│                                                     │
│  Step 3: Replace in Text                               │
│     └── Output: "Thanks John for the update on Sarah's     │
│              ticket"                                   │
│                                                     │
└──────────────────────────────────────────────────────────────
───────────────────────────┘
```

API Methods Used

Method	Format	Purpose
conversations.history	JSON	Get channel messages
conversations.replies	Form-urlencoded	Get thread replies
users.info	Form-urlencoded	Resolve user ID to name
conversations.join	JSON	Join public channel
chat.postMessage	JSON	Post messages

Key Discussions Feature

The "Key Discussions" feature combines both Slack messages and Linear comments to provide a comprehensive view of project discussions:

```
┌──────────────────────────────────────────────────────────────────────────
──────────────────────────┐
│          KEY DISCUSSIONS OUTPUT                     │
├──────────────────────────────────────────────────────────────────────────
──────────────────────────┤
│                                        │
│  ### Key Discussions                             │
│                                        │
│  1. **PRD Review Meeting** (Slack - Jan 25)             │
│    John raised concerns about the API design...          │
│                                        │
│  2. **Blocker: Database Migration** (Linear - Jan 24)    │
│    Sarah commented: "We need to resolve the schema        │
│    conflicts before proceeding..."                  │
│                                        │
│  3. **Design Sync** (Slack - Jan 23)                │
│    Team discussed the new UI components...             │
│                                        │
└──────────────────────────────────────────────────────────────────────────
──────────────────────────┘
```

LLM Query Interpreter

Overview

The LLM Query Interpreter uses the Fuelix API to understand natural language queries, correct typos, and extract intent.

Intent Types

Intent	Description	Example Queries
pod_info	Information about a specific pod	"whats up with fts", "how is talent studio"
project_info	Information about a project	"AI interviewer status", "deep dive cohorts"
all_pods	Overview across all pods	"how are all teams doing", "overall status"
unknown	Cannot determine intent	"hello", ambiguous queries

Query Processing Flow

```
┌─────────────────────────────────────────────────────────────
─────────────────────┐
│           LLM QUERY INTERPRETER                   │
├─────────────────────────────────────────────────────────────
─────────────────────┤
│                                                   │
│  Input: "whats going on AI Interviwer"            │
│                                                   │
│  Step 1: Send to LLM with context                 │
│    └── Available pods list                        │
│    └── Sample project names                       │
│                                                   │
│  Step 2: LLM Response (JSON)                      │
│   {                                               │
│     "intent": "project_info",                     │
│     "entity": "AI Interviewer",    // Typo corrected   │
│     "confidence": "high",                         │
│     "corrected_query": "What's going on with AI        │
│                    Interviewer project?"          │
│   }                                               │
│                                                   │
│  Step 3: Route to appropriate handler             │
│    └── project_info → getLiveProject()            │
│                                                   │
└─────────────────────────────────────────────────────────────
─────────────────────┘
```

Confidence Levels

- **High**: Exact match or clear intent (>90% confidence)
- **Medium**: Fuzzy match with context clues (60-90%)
- **Low**: Ambiguous or unclear query (<60%)

Historical Data & Context Capture

Overview

Two services work together to capture qualitative and quantitative data for Q1 retrospective analysis:

1. **Historical Data Service**: Weekly snapshots and cycle closings
2. **Context Capture Service**: Linear comments, Slack messages, project updates

Historical Data Service (historicalDataService.js)

```
┌──────────────────────────────────────────────────────────────────────────
─────────────────────────┐
│         HISTORICAL DATA SERVICE                     │
├──────────────────────────────────────────────────────────────────────────
─────────────────────────┤
│                                                     │
│  Tables:                                            │
│   ├── weekly_snapshots: Time-series metrics (weekly)       │
│   ├── cycle_closings: End-of-cycle summaries               │
│   ├── events_log: Notable events, blockers, wins           │
│   └── quarterly_goals: Q1 goals vs actuals                 │
│                                                     │
│  Functions:                                         │
│   ├── captureWeeklySnapshot(cycleKpi, featureMovement)     │
│   ├── closeCycle(cycle, pod, metrics, notes)               │
│   ├── logEvent(type, title, description, impact)           │
│   ├── setQuarterlyGoal(pod, goalType, target)              │
│   ├── getWeeklyTrend(pod, metric, weeks)                   │
│   └── getCycleComparison(pod, cycles)                      │
│                                                     │
└──────────────────────────────────────────────────────────────────────────
─────────────────────────┘
```

Context Capture Service (contextCaptureService.js)

```
┌──────────────────────────────────────────────────────────────────────────
─────────────────────────┐
│         CONTEXT CAPTURE SERVICE                     │
├──────────────────────────────────────────────────────────────────────────
─────────────────────────┤
│                                                     │
│  Purpose: Capture "WHY" behind the numbers                 │
│                                                     │
│  Tables:                                            │
│   ├── linear_comments: Issue comments with sentiment       │
│   ├── project_updates: Status changes, descriptions        │
│   ├── slack_messages: Team channel discussions             │
│   └── del_issues: DEL issue details (beyond counts)        │
│                                                     │
│  Classification:                                    │
│   ├── sentiment: positive/negative/neutral                 │
│   ├── isBlocker: true/false                                │
│   ├── isRisk: true/false                                   │
│   ├── isDecision: true/false                               │
│   └── keywords: extracted topics                           │
│                                                     │
└──────────────────────────────────────────────────────────────────────────
─────────────────────────┘
```

Weekly Snapshot Schema

Field	Type	Description

snapshot_date	TEXT	YYYY-MM-DD
week_number	INTEGER	Week of year
cycle	TEXT	C1-C6
pod	TEXT	Pod name
committed_del	INTEGER	DELs committed
completed_del	INTEGER	DELs completed
delivery_pct	INTEGER	Delivery percentage
spillover	INTEGER	Spillover count
planned_features	INTEGER	Total projects
features_done	INTEGER	Completed projects
features_in_flight	INTEGER	Started projects
features_not_started	INTEGER	Backlog projects

Caching Strategy

Cache Architecture

```
┌─────────────────────────────────────────────────────────────────
─────────────────────────┐
│                    CACHE SYSTEM                             │
├─────────────────────────────────────────────────────────────────
─────────────────────────┤
│                                                    │
│  API Cache (File-based):                                │
│   ├── Location: agent/output/cache/                      │
│   ├── Format: JSON files with MD5 hash keys               │
│   └── TTL Configuration:                               │
│       ├── Projects:  5 minutes (300,000 ms)             │
│       ├── Issues:    3 minutes (180,000 ms)             │
│       ├── Comments:  2 minutes (120,000 ms)              │
│       └── Default:   5 minutes                         │
│                                                    │
│  Dashboard Cache (localStorage):                         │
│   ├── Location: Browser localStorage                     │
│   ├── Key: kpi_dashboard_cache                           │
│   └── Purpose: Instant dashboard loading                  │
│                                                    │
│  Cache Entry Structure:                                 │
│  {                                                  │
│    "data": <cached response>,                          │
│    "expiresAt": <timestamp ms>                          │
│  }                                                  │
│                                                    │
└─────────────────────────────────────────────────────────────────
─────────────────────────┘
```

Cache Decorator Pattern

```
// withCache(key, asyncFn, ttl) → Cached async function
const fetchFn = withCache(
  `projects_${initiativeId}`,
  async () => client.getProjectsByInitiative(initiativeId),
  CACHE_TTL.projects
);

const projects = await fetchFn();  // Returns cached or fresh data
```

Cache Flow

```
Request
   │
   ▼
┌─────────────────┐     ┌─────────────────┐
│  Check Cache    │────▶│  Cache HIT      │────▶ Return cached data
│  (file)         │     │  & not          │
└─────────────────┘     │  expired        │
   │                    └─────────────────┘
   │ MISS or EXPIRED
   ▼
┌─────────────────┐
│  Fetch from     │
│  Linear API     │
└─────────────────┘
   │
   ▼
┌─────────────────┐
│  Store in       │────▶ Return fresh data
│  Cache          │
└─────────────────┘
```

Configuration Management

Configuration Files

```
config/
├── linear_ids.json      # Primary config (auto-generated)
├── pods.json            # Manual pod config (fallback)
├── cycle_calendar.json  # Cycle date ranges
└── label_ids.json       # Label ID mappings (auto-generated)
```

Config Load Priority

```
┌──────────────────────────────────────────────────────────────────
──────────────────┐
│          CONFIG LOADING PRIORITY                    │
├──────────────────────────────────────────────────────────────────
──────────────────┤
│                                                     │
│  1. config/linear_ids.json (PREFERRED)              │
│     └── Contains: org, pods with initiativeId, projects   │
│                                                     │
│  2. config/pods.json (FALLBACK)                     │
│     └── Contains: minimal pod config with teamId         │
│                                                     │
│  Auto-generated by runWeeklyKpi.js:                 │
│   ├── config/linear_ids.json                        │
│   └── config/label_ids.json                         │
│                                                     │
└──────────────────────────────────────────────────────────────────
──────────────────┘
```

Pod Configuration Schema

```
// config/linear_ids.json
{
  "org": {
    "id": "org-uuid",
    "name": "Telus Digital AI Engineering",
    "urlKey": "playment"
  },
  "pods": {
    "FTS": {
      "teamId": "team-uuid",
      "initiativeName": "Q1 2026 - FTS Roadmap",
      "initiativeId": "initiative-uuid",
      "projects": [
        {
          "id": "project-uuid",
          "name": "Q1 2026 : Feature Name",
          "state": "started"
        }
      ]
    },
    "ML": {
      "teamId": "ml-team-uuid",
      "initiativeName": "Q1 2026 - ML Roadmap",
      "initiativeId": "ml-initiative-uuid",
      "projects": []
    },
    "FOT": {
      "teamId": "fot-team-uuid",
      "initiativeName": "Q1 2026 - FOT Roadmap",
      "initiativeId": "fot-initiative-uuid",
      "projects": [...]
    },
    "BTS": {
      "teamId": "bts-team-uuid",
      "initiativeName": "Q1 2026 – BTS Roadmap",
      "initiativeId": "bts-initiative-uuid",
      "projects": []
    }
    // ... more pods
  }
}
```

Cycle Calendar Schema

```
// config/cycle_calendar.json
{
  "pods": {
    "FTS": {
      "C1": { "start": "2026-01-06", "end": "2026-01-20" },
      "C2": { "start": "2026-01-20", "end": "2026-02-03" },
      "C3": { "start": "2026-02-03", "end": "2026-02-17" },
      "C4": { "start": "2026-02-17", "end": "2026-03-03" },
      "C5": { "start": "2026-03-03", "end": "2026-03-17" },
      "C6": { "start": "2026-03-17", "end": "2026-03-31" }
    },
    "ML": { ... },
    "FOT": { ... },
    "BTS": { ... }
    // ... more pods
  }
}
```

API Reference

Linear GraphQL Queries

```
# Fetch issues with DEL label for a team
query IssuesByTeamAndLabel($teamId: ID!, $labelId: ID!, $first: Int!, $after: String) {
  issues(first: $first, after: $after, filter: {
    team: { id: { eq: $teamId } },
    labels: { id: { eq: $labelId } }
  }) {
    nodes {
      id
      identifier
      createdAt
      completedAt
      state { type name }
      labels { nodes { id name } }
    }
    pageInfo { hasNextPage endCursor }
  }
}

# Fetch projects by initiative
query ProjectsByInitiative($initiativeId: ID!, $first: Int!, $after: String) {
  projects(first: $first, after: $after, filter: {
    initiatives: { id: { eq: $initiativeId } }
  }) {
    nodes {
      id
      name
      state
      lead { name }
      targetDate
      url
      updatedAt
    }
    pageInfo { hasNextPage endCursor }
  }
}
```

CLI Commands Reference

Command	Description	Handler
kpi	Combined DEL + Feature Movement	all_pods_summary
weekly kpi	Same as kpi	all_pods_summary
what's going on across all pods	Cross-pod summary	all_pods_summary
what's happening this week	Cross-pod summary	all_pods_summary
pods	List all pods	list_pods
pod <name>	Pod summary	pod_summary
pod <name> projects	List pod projects	pod_projects
what's going on in <pod>	Rich pod narrative	pod_narrative
project <name>	Project details	project_detail
deep dive <project>	Full analysis (Slack+Linear)	project_deep_dive
project <name> blockers	Show blockers	project_blockers

project <name> comments	Comment summary	project_comments
DELs in C1	DELs planned for cycle	dels_by_cycle
pending dels	Show incomplete DELs	pending_dels
del kpi	DEL-only KPI table	del_kpi
debug	Toggle debug mode	debug_mode
clear cache	Clear API cache	clear_cache
/refresh	Regenerate snapshot	Internal
/help	Show help	Internal

Database Schema

SQLite Schema (state/kpi_state.db)

```
-- Snapshot issue tracking
CREATE TABLE snapshots (
  pod TEXT NOT NULL,
  cycle TEXT NOT NULL,
  issueId TEXT NOT NULL,
  PRIMARY KEY (pod, cycle, issueId)
);

-- Snapshot metadata
CREATE TABLE snapshot_meta (
  pod TEXT NOT NULL,
  cycle TEXT NOT NULL,
  frozen INTEGER NOT NULL DEFAULT 0,
  frozenAt TEXT,
  lastRefreshAt TEXT,
  committedCount INTEGER NOT NULL DEFAULT 0,
  PRIMARY KEY (pod, cycle)
);

-- Weekly snapshots (historical)
CREATE TABLE weekly_snapshots (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  snapshot_date TEXT NOT NULL,
  week_number INTEGER NOT NULL,
  cycle TEXT NOT NULL,
  pod TEXT NOT NULL,
  committed_del INTEGER DEFAULT 0,
  completed_del INTEGER DEFAULT 0,
  delivery_pct INTEGER DEFAULT 0,
  spillover INTEGER DEFAULT 0,
  planned_features INTEGER DEFAULT 0,
  features_done INTEGER DEFAULT 0,
  features_in_flight INTEGER DEFAULT 0,
  features_not_started INTEGER DEFAULT 0,
  created_at TEXT DEFAULT (datetime('now')),
  UNIQUE(snapshot_date, pod, cycle)
```

```
);

-- Cycle closings
CREATE TABLE cycle_closings (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  cycle TEXT NOT NULL,
  pod TEXT NOT NULL,
  closed_at TEXT NOT NULL,
  final_committed INTEGER DEFAULT 0,
  final_completed INTEGER DEFAULT 0,
  final_delivery_pct INTEGER DEFAULT 0,
  final_spillover INTEGER DEFAULT 0,
  features_completed INTEGER DEFAULT 0,
  features_total INTEGER DEFAULT 0,
  went_well TEXT,
  went_wrong TEXT,
  notes TEXT,
  created_at TEXT DEFAULT (datetime('now')),
  UNIQUE(cycle, pod)
);

-- Events log
CREATE TABLE events_log (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  event_date TEXT NOT NULL,
  cycle TEXT,
  pod TEXT,
  event_type TEXT NOT NULL,
  title TEXT NOT NULL,
  description TEXT,
  impact TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

-- Linear comments capture
CREATE TABLE linear_comments (
  id TEXT PRIMARY KEY,
  issue_id TEXT NOT NULL,
  issue_identifier TEXT,
  issue_title TEXT,
  project_id TEXT,
  project_name TEXT,
  pod TEXT,
  cycle TEXT,
  author TEXT,
  body TEXT,
  created_at TEXT,
  captured_at TEXT DEFAULT (datetime('now')),
  sentiment TEXT,
  is_blocker INTEGER DEFAULT 0,
  is_risk INTEGER DEFAULT 0,
  is_decision INTEGER DEFAULT 0,
```

```sql
  keywords TEXT
);

-- Slack messages capture
CREATE TABLE slack_messages (
  id TEXT PRIMARY KEY,
  channel_id TEXT,
  channel_name TEXT,
  pod TEXT,
  cycle TEXT,
  author TEXT,
  text TEXT,
  thread_ts TEXT,
  ts TEXT,
  created_at TEXT,
  captured_at TEXT DEFAULT (datetime('now')),
  sentiment TEXT,
  is_blocker INTEGER DEFAULT 0,
  is_risk INTEGER DEFAULT 0,
  is_announcement INTEGER DEFAULT 0,
  keywords TEXT
);

-- Quarterly goals
CREATE TABLE quarterly_goals (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  quarter TEXT NOT NULL,
  pod TEXT NOT NULL,
  goal_type TEXT NOT NULL,
  target_value INTEGER,
  target_description TEXT,
  actual_value INTEGER,
  status TEXT DEFAULT 'pending',
  created_at TEXT DEFAULT (datetime('now')),
  UNIQUE(quarter, pod, goal_type)
);
```

Freeze Policy

```
┌──────────────────────────────────────────────────────────────────────
─────────────────┐
│              FREEZE POLICY                       │
├──────────────────────────────────────────────────────────────────────
─────────────────┤
│                                         │
│  C1, C2 snapshots:                               │
│    └── Freeze after end of C2 (adoption grace period)     │
│                                         │
│  C3, C4, C5, C6 snapshots:                         │
│    └── Freeze at their own cycle end                   │
│                                         │
│  While not frozen:                              │
│    └── Snapshots can be refreshed (late labeling OK)      │
│                                         │
│  Once frozen:                                 │
│    └── committedCount becomes permanent                 │
│                                         │
└──────────────────────────────────────────────────────────────────────
─────────────────┘
```

Environment Variables

Variable	Required	Default	Description
LINEAR_API_KEY	Yes	-	Linear API authentication
LINEAR_GQL_URL	No	https://api.linear.app/graphql	GraphQL endpoint
SLACK_BOT_TOKEN	No	-	Slack Bot OAuth token
FUELIX_API_KEY	No	-	LLM API key
FUELIX_API_URL	No	https://api.fuelix.ai	LLM endpoint
FUELIX_MODEL	No	gpt-5.2	LLM model name
SNAPSHOT_PATH	No	agent/output/latest_snapshot.json	Snapshot file path
KPI_CYCLE	No	Auto-detect	Override cycle (C1-C6)
FREEZE_POLICY_CYCLE	No	C2	When C1/C2 freeze

Output Artifacts

Agent Outputs

```
agent/output/
├── latest_snapshot.json    # Point-in-time KPI snapshot
├── answers.log.jsonl       # Q&A audit log
├── cache/                  # API response cache
│   └── *.json
└── weekly_agent_summary.md # Generated summaries
```

Script Outputs

```
out/
├─── kpi_weekly_report.md    # Markdown KPI report
├─── pod_cycle_kpi.csv       # DEL metrics CSV
└─── pod_feature_movement.csv # Feature movement CSV
```

State Directory

```
state/
├─── kpi_state.db            # SQLite database
└─── history/                # Historical exports
```

NPM Scripts

```
# Generate snapshot from Linear
npm run snapshot

# Generate with debug output
npm run debug:snapshot

# Start interactive CLI
npm run agent

# Run full weekly KPI (requires LINEAR_API_KEY env)
LINEAR_API_KEY=<key> node scripts/runWeeklyKpi.js

# Capture context for Q1 analysis
node scripts/captureContext.js

# Log notable events
node scripts/logEvent.js

# Set quarterly goals
node scripts/setGoals.js

# Analyze Q1 retrospective
node scripts/analyzeQ1.js
```

Error Handling

Error Types

Error Code	Description	Resolution
MISSING_LABEL_IDS	label_ids.json not found	Run runWeeklyKpi.js
MISSING_CYCLE_CALENDAR	cycle_calendar.json missing	Create config file
POD_NOT_FOUND	Invalid pod name	Check available pods
PROJECT_NOT_FOUND	Project not in any pod	Verify project name
NO_INITIATIVE_ID	Pod missing initiativeId	Update linear_ids.json
FETCH_FAILED	API request failed	Check API key/network

SLACK_AUTH_ERROR	Slack token invalid	Check SLACK_BOT_TOKEN
SLACK_SCOPE_ERROR	Missing Slack scopes	Add required scopes to app

---

Summary

This system provides a comprehensive KPI tracking solution with:

1. **Real-time Integration**: Live Linear API queries with intelligent caching
2. **Historical Tracking**: SQLite-based snapshot storage with freeze policies
3. **Flexible Querying**: Natural language + structured command support
4. **LLM Query Interpreter**: Typo correction and intent extraction
5. **Dual KPI System**: DEL metrics + Feature Movement tracking
6. **Cross-Pod Search**: Fuzzy project matching across all 9 pods
7. **Slack Integration**: Team discussions with user ID resolution
8. **Key Discussions**: Combined Slack + Linear comment summaries
9. **Project Deep Dive**: Comprehensive project analysis
10. **Beautiful CLI Output**: Unicode box-drawing tables via tableFormatter.js
11. **Cross-Pod Summaries**: "What's happening across all pods" queries
12. **Pending DEL Tracking**: View incomplete deliverables by pod
13. **Historical Analysis**: Q1 retrospective data capture and trends
14. **Dashboard Caching**: Instant loading with localStorage
15. **Feature Readiness**: Track PRD/Design/Dev phases per project

The architecture follows a layered approach with clear separation between CLI, business logic, data access, and external services.

Document last updated: January 2026