# Talend-to-dbt Migration Agent

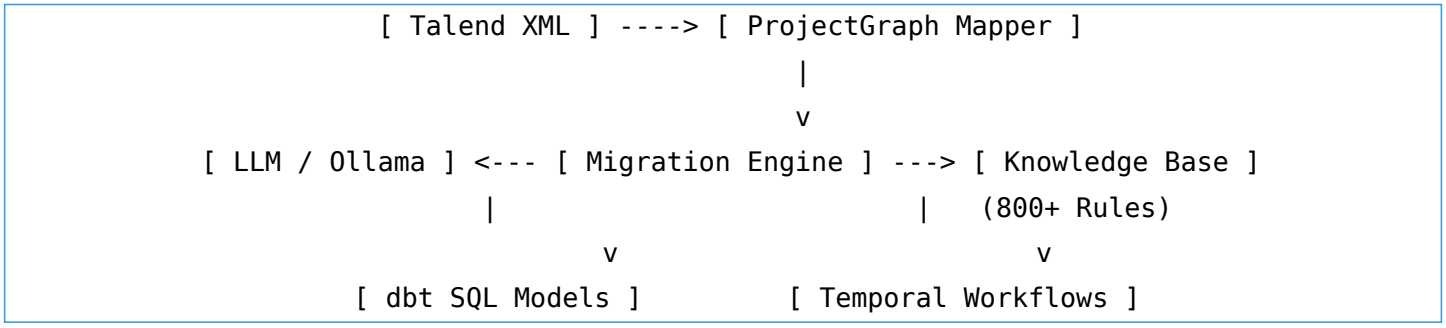
Technical Specification & Operational Manual

---

Version 1.0.1 | Optimized for DuckDB & Temporal.io

# 1. System Architecture

The agent utilizes a decoupled architecture to separate metadata parsing from logical code generation.




## 1.1 Core Components

- Knowledge Base: More than 800+ rules for components and full Java Routine translation.

- Migration Engine: Parallel multi-threaded processing optimized for i7-14700K.

- Sourav Agent: Context-aware inference engine using local LLMs via Ollama.

# 2. Project Framework

## 2.1 Directory Structure

```
talendtodbtsouravagent/
├── input_data/           # Source Talend .item files
├── output/
│   ├── models/          # Generated dbt SQL models
│   ├── macros/          # Reusable Joblet macros
│   └── temporal/        # Python workflow files
├── src/
│   ├── main_engine.py   # Orchestration logic
│   ├── agent_llm.py     # LLM interface
│   └── knowledge_base.py# Rule Encyclopedia
└── run_migration.py     # Entry point
```

# 3. Installation & Optimization

## 3.1 Environment Setup

1. Install Dependencies: pip install -r requirements.txt

2. Configure Ollama: Ensure 'llama3' or 'mistral' is pulled and 'ollama serve' is active.

## 3.2 Hardware Saturation (RTX 5070)

The agent is configured to squeeze maximum performance from 12GB VRAM:

```
num_ctx   = 4096  # VRAM safe limit - increase if needed
num_gpu   = 999   # 100% offload
num_thread= 8     # P-core target
```

# 4. Operational Guidelines

Run 'python run_migration.py' to begin. The engine will topologically sort components and generate valid CTE chains.

## 4.1 Sample Input (Talend XML)

```
...
```

## 4.2 Sample Output (dbt SQL)

```
WITH tFileInput_1 AS (
    SELECT * FROM {{ source('raw', 'data') }}
),
tMap_1 AS (
    SELECT col1, upper(col2) FROM tFileInput_1
),
final_cte AS (
    SELECT * FROM tMap_1
)
SELECT * FROM final_cte;
```