DishTV AIP Dashboard System

Reverse Engineering, Architecture Assessment & Product Requirements Document

System type	AI-assisted automated operational intelligence & management reporting
Production scope	DishTV Unified, ShopZop, VZY/OTT; TV Business planned
Core stack	Claude • n8n • LangChain • Tursio MCP • Azure • Python • Chart.js • Microsoft Graph • Prometheus
Primary delivery	Scheduled self-contained HTML dashboard distributed through email
Assessment basis	Supplied project instructions, live dispatcher source, frontend engine and Service Insight Agent workflow JSON

Current-state reverse engineering pack • v0.1

Prepared from the supplied production context and source artifacts. Where implementation evidence is incomplete, this document explicitly marks the item as inferred, pending validation, or planned.

1. Executive Summary

The DishTV AIP Dashboard System is an automated operational intelligence platform that combines n8n workflow agents, Tursio MCP natural-language-to-SQL data access, Azure-hosted LLM analysis, Python batch orchestration and interactive HTML dashboards.

The production pattern is a scheduled batch pipeline: KPI agents retrieve structured operational data, family-level insight agents interpret the data, a Meta Insight Agent produces a cross-family executive narrative, and a Python dispatcher packages the results into a self-contained HTML dashboard that is distributed through Microsoft Graph email.

Key architectural distinction: the Claude Project shown in the supplied screenshot is best understood as the AI-assisted development/control workspace. The production runtime is distributed across n8n, Tursio MCP, Azure services, a dispatcher VM and Microsoft Graph.

Current-state assessment: production maturity is high for scheduled reporting and workflow automation, with good retry/failure handling and observability. Current limitations include static/batch-generated insight text after drill-down, documentation drift around KPI counts, pending Call Center narrative wiring, and an unstarted TV Business dashboard.

2. Reconstructed System Architecture

End-to-end flow

Claude development workspace → n8n workflow orchestration → KPI agents → Tursio MCP NL→SQL → operational data → structured KPI JSON → family insight agents → Claude Sonnet through Azure → Meta Insight → Python dispatcher → DASH_DATA → HTML/CSS/Chart.js frontend → Microsoft Graph → management email.

Layer	Primary component	Responsibility
Development	Claude Project	AI-assisted workflow/code development, troubleshooting and project context management
Orchestration	n8n	Runs KPI and insight agent workflows exposed through webhooks
Data access	Tursio MCP	Natural-language-to-SQL interface to KPI data sources
Reasoning	n8n LangChain + Claude Sonnet 4.6	Converts KPI data into analytical commentary
Batch backend	Python dispatcher VM	Fetch, retry, aggregate, render, monitor and distribute
Presentation	HTML/CSS/JavaScript + Chart.js	Interactive tabs, charts, filters and drill-down
Distribution	Microsoft Graph	Sends dashboard HTML as email attachment
Observability	Prometheus + NDJSON/Filebeat/EKB	Run health, KPI status and execution records
Secrets	Azure Key Vault + Managed Identity	Secure Graph credential retrieval

3. Production Product Portfolio

Product	Schedule	Current scope
DishTV Unified	6:00 AM IST	Business, Retention, Sales, Service, Call Center, Per ASE/Lakh
ShopZop	7:00 AM IST	13 KPI cards
VZY/OTT	7:30 AM IST	13 KPI cards; internally uses OTT keys, display renamed to VZY
TV Business Dashboard	Not started	Planned 13 KPI agents using TV_AI MCP, dispatcher and frontend

The supplied production dispatcher specifically implements the DishTV Unified daily pipeline. ShopZop and VZY/OTT are documented as parallel production dashboard products but their full source artifacts were not supplied for this audit.

4. KPI & Agent Inventory

The current **KPI_AGENTS** list in the supplied dispatcher contains 37 configured KPI integrations. The project instructions state 36, creating a documentation/configuration discrepancy that should be reconciled.

Family	KPI agents in current dispatcher
Business / Retention	Revenue; Active Base; Gross Adds; Net Adds; ARPU; CSAT T2-B2; LTR; FMR; Recharge; PBR; RADA; Borndead; Recharg
Sales	Gross Adds; Active DS; Tran 1; UAD; Borndead; PBR; Non-Trade Sales; 999 Primary & Secondary Sales
Service	24HR FR TAT; 4HR FR TAT; Short HR TAT; 24HRS Pending; FR M-2; FR Repeat; Happy Code; EQI; Call Cancellation; Cal
Call Center	Inbound Quality; Inbound CSAT; Fatal Calls; Abandoned Ratio; Service Level; Complaint Ratio

5. AI Insight Architecture

The insight layer answers a different question from the KPI layer. KPI agents establish **what happened**; insight agents are prompted to determine **what the movement means operationally**.

Agent family	Endpoint / role	Output
Business	/webhook/dashboard-insight-agent	Business KPI insights + combined family insight
Retention	/webhook/retention-insight-agent	Retention KPI insights + combined family insight
Sales	/webhook/sales-insight-agent	Sales KPI insights + combined family insight
Service	/webhook/service-insight-agent	11 service KPI insights + combined service-quality insight
Call Center	/webhook/callcenter-insight-agent	Call Center KPI insights + combined family insight
Meta	/webhook/meta-insight-agent	Cross-family executive synthesis

6. Reverse Engineering: Service Insight Agent

The supplied workflow JSON provides a complete example of the insight-agent design pattern.

Workflow chain

Service Insight Webhook → Input Validator → Generate Service Insights → Output Shaper. The Azure OpenAI Chat Model is connected to the LangChain agent as its ai_languageModel dependency.

Node	Function
Service Insight Webhook	POST endpoint at /webhook/service-insight-agent; receives scope, drill context and KPI payloads
Input Validator	Normalizes input, resolves scope label and converts KPI JSON into a controlled analytical text representation
Generate Service Insights	LangChain agent with explicit operational KPI interpretation rules and structured JSON output contract
Azure OpenAI Chat Model	Claude Sonnet 4.6 via Azure; maxTokens 4000; temperature 0.3
Output Shaper	Removes code fences, parses JSON, attempts fallback extraction and merges safe defaults

Service geography: the workflow explicitly recognizes eight service zones: NORTH 1, NORTH 2, EAST 1, EAST 2, SOUTH 1, SOUTH 2, WEST 1, WEST 2, with Customer Service Managers beneath them. This is intentionally distinct from the standard four-zone sales geography.

Analytical prompt design: the agent is told to detect trajectory, inflection points, cross-KPI relationships, zone/CSM outliers, MTD run-rate effects and material EQI movement. It is instructed to avoid data restatement and keep each KPI insight to 2–3 sentences.

7. Reverse Engineering: Python Dispatcher

The Python dispatcher is the production batch orchestrator and the central integration point between n8n, the dashboard frontend, monitoring and email distribution.

Phase	Action
Startup	Parse CLI options and load Azure Key Vault secrets using VM Managed Identity
1	Fetch all configured KPI agents with retry and failure isolation
2	Fetch family-level insight agents using available KPI payloads
2.5	Fetch cross-family Meta Insight using KPI payloads and family insights
3	Build DASH_DATA contract
4	Render self-contained HTML from CSS, body, Chart.js and engine.js
Observability	Write Prometheus textfile metrics and NDJSON execution records
5	Send dashboard through Microsoft Graph

Retry policy: requests retry every 30 seconds for up to 15 minutes per call, with a 180-second per-request timeout. Permanent HTTP failures such as 400, 401, 403, 404, 405, 410 and 415 are not retried.

Failure isolation: an individual KPI failure does not prevent the report from being rendered. If every KPI fails, the dispatcher aborts and attempts a panic email.

8. Reverse Engineering: Dashboard Frontend

Component	Role
FAMILY_META	Defines KPI order, titles, chart types, data fields, divisors, decimal precision and insight keys
getRows()	Slices KPI data based on All India, Zone or Circle/CSM state
attachDrill() / drillDown()	Implements double-click drill-down
goUp()	Implements right-click navigation back to higher scope
renderAbsolute()	Absolute values with optional MTD projection
renderSignedAbs()	Signed absolute KPI with positive/negative treatment
renderNormal()	Percentage KPI
renderSigned()	Signed percentage KPI
renderGrouped()	M1/M3 comparison
renderGrouped2Series()	Two-series counts such as UAD1/UAD2 and 999 Primary/Secondary
renderStacked100()	100% distribution charts such as LTR, FMR and Recharge
injectInsights()	Loads batch-generated insights from DASH_DATA into page narrative elements

Interactive hierarchy: All India → Zone for selected month → Circle/CSM for selected month. Service uses its separate eight-zone/CSM taxonomy. Call Center is explicitly All-India only and hides the geographic filter.

Important current limitation: insight injection occurs once at page load. If a user drills into a zone or CSM, charts change but the narrative insight remains based on the batch-generated All-India view unless additional asynchronous scope-specific insight calls are introduced.

9. DASH_DATA Contract

The dispatcher creates a single JSON object and embeds it in the generated page as **window.DASH_DATA**. The frontend treats this as its primary data contract.

Key fields: scope_type, scope_value, scope_label, recipient_name, current_date, filter_enabled, zones_list, circles_by_zone, service_zones_list, service_circles_by_zone, kpi_data, insights, meta_insight and failed_kpis.

Design characteristic: after the morning batch completes, the dashboard is self-contained. The browser does not need to query the KPI backend simply to render the delivered report.

10. Deployment & Security Model

Area	Current implementation
Runtime VM	localusr@10.92.74.15; dispatcher under /home/dispatcher/
Python environment	/home/dispatcher/dispatcher-venv/
Scheduling	Linux cron; DishTV Unified documented at 30 0 * * * (06:00 IST)
n8n	https://n8n.dishtv.in, also documented as prod-n8n.dishtv.in
Secrets	Azure Key Vault retrieved using VM system-assigned Managed Identity
LLM credential	n8n Azure Open AI account credential
Email	Microsoft Graph client-credentials flow
Frontend assets	body.html, engine.js, styles.css, chartjs.min.js

11. Observability & Operational Controls

Mechanism	Signals / purpose
Prometheus textfile collector	Run start/finish timestamps, total KPI success/failure, per-KPI status, per-family insight status
NDJSON results	One machine-readable record per KPI and insight execution, including elapsed time
Daily logs	Dispatcher log file under /var/log/dispatcher/
Panic email	Notifies owner when all KPIs fail, rendering fails, or major delivery errors occur

12. Key Technical Decisions Captured

Decision	Reason / implication
LangChain agent typeVersion 3.1	Required by the project context to avoid the conversationalAgent bug associated with 1.7
Azure model as top-level peer node	Supports the n8n wiring pattern required for credentials and agent model connection
Correct Tursio table name in prompt prefix	Project notes indicate Tursio NL→SQL can rewrite queries if the correct table name is not explicitly referenced
Merge node for 999 fan-in	Uses append mode to avoid race conditions
Separate sales vs service geography	Prevents incompatible zone/circle taxonomies from being reused
QRC month format MM-YYYY	Required data-format detail captured in project context
n8n SDK credential limitation	Workflow JSON must be imported through UI to carry credentials for certain insight agents

13. Current-State Gaps & Risks

Item	Current state	Priority
KPI count discrepancy	Instructions say 36; supplied dispatcher config contains 37	Medium
Call Center narrative injection	Call Center insight agent is registered, but project notes state body.html placeholders still need dynamic injection	High

Item	Current state	Priority
Scope-specific insights	Charts support drill-down; insight text is batch-generated and remains at All-India conte	Medium
engine.js patch v2	Pending change to by_zone handling while preserving Call Center interaction restrictions	Medium
TV Business Dashboard	13 KPI agents + TV_AI + dispatcher + frontend not started	Planned
Documentation drift	Project context contains historical notes mixed with current implementation details	Medium

14. Product Requirements Document

Product name: DishTV AIP Dashboard System

Product type: AI-assisted automated operational intelligence and executive reporting platform.

Primary objective: transform operational KPI data into daily management dashboards containing historical trends, geographic analysis, AI-generated commentary, family-level interpretation and cross-domain executive synthesis.

Primary users: business leadership, operations leadership, service leadership, sales leadership, retention teams, call center leadership and management stakeholders.

ID	Functional requirement
FR-01	Retrieve KPI data automatically on a scheduled basis.
FR-02	Provide six-month historical KPI context for the configured dashboard.
FR-03	Support All India scope.
FR-04	Support zone-level analysis where the KPI geography permits.
FR-05	Support circle/CSM-level analysis where the KPI geography permits.
FR-06	Render KPI-specific visualizations appropriate to the data shape.
FR-07	Generate KPI-level analytical insights.
FR-08	Generate family-level combined insights.
FR-09	Generate cross-family executive synthesis.
FR-10	Continue report generation when individual KPI agents fail.
FR-11	Notify the owner when the complete KPI acquisition fails.
FR-12	Distribute the generated dashboard automatically.
FR-13	Publish operational execution telemetry.
FR-14	Store machine-readable execution results.

Non-functional requirements

Area	Requirement
Reliability	Retry transient failures for up to 15 minutes per call and isolate individual KPI failures.
Security	Do not hard-code Graph secrets; use Azure Key Vault and managed identity.
Observability	Expose run and component health through Prometheus and structured execution records.
Maintainability	Keep KPI retrieval, insight generation, dispatcher, templates and frontend rendering separated.
Extensibility	Allow new KPI and insight agents to be registered through configuration structures.
Delivery	Produce a self-contained HTML dashboard suitable for email distribution and browser viewing.

15. Resume & Portfolio Positioning

Recommended project title: DishTV AI Operational Intelligence Platform

Recommended positioning: AI-assisted automated KPI and management reporting platform integrating n8n agents, MCP-based data retrieval, Python orchestration, Azure-hosted LLMs and interactive HTML dashboards.

For a resume, contribution language should be aligned to the work personally performed. The supplied evidence establishes the production architecture and current capabilities, but does not by itself establish that one individual authored the complete platform.

- Automated 37 configured operational KPI integrations across Business, Retention, Sales, Service and Call Center domains with six-month trend analysis and geographic drill-down.
- Implemented AI-generated KPI, domain and executive-level analysis using n8n LangChain agents and Claude Sonnet through Azure.
- Managed batch orchestration, retry handling, failure isolation, Azure Key Vault secrets, Microsoft Graph distribution and Prometheus/NDJSON monitoring.
- Supported daily management reporting through self-contained interactive dashboards with All India → Zone → Circle/CSM analysis.

Interview explanation: "It is an automated operational intelligence platform. n8n KPI agents retrieve structured operational data through a Tursio MCP natural-language-to-SQL layer. A Python dispatcher orchestrates the calls, handles retries and failures, invokes domain-specific AI insight agents, generates a self-contained HTML dashboard and distributes it through Microsoft Graph. The frontend supports geographic drill-down, while Prometheus and NDJSON provide operational monitoring."

16. Technical Assessment

Dimension	Assessment	Basis
Business maturity	Production	Scheduled daily reporting across multiple dashboard products
Automation	High	Cron + n8n agents + dispatcher + automated email
AI usage	Medium-High	KPI interpretation, family insights and Meta Insight synthesis
Workflow orchestration	High	n8n webhook agent registry and dispatcher coordination
Reliability	Good	15-minute retry, failure isolation and panic handling
Monitoring	Good	Prometheus metrics + NDJSON + logs
Frontend sophistication	Medium	Interactive Chart.js dashboard with drill-down
Real-time capability	Low	Batch-generated dashboard and one-shot insight injection
Extensibility	Good	Config-driven KPI and insight agent registration
Documentation consistency	Needs cleanup	36 vs 37 KPI discrepancy and evolving pending-work notes

17. Current-State Blueprint

```
06:00 IST → Linux Cron → Python Dispatcher → KPI Webhooks → n8n Agents → Tursio MCP → Operational Data →
KPI JSON → Family Insight Agents → Claude Sonnet 4.6 via Azure → Meta Insight → DASH_DATA →
HTML/CSS/Chart.js/engine.js → Prometheus + NDJSON → Microsoft Graph → Management.
```

Architectural characterization: the strongest description supported by the supplied artifacts is **AI-assisted operational intelligence and automated management reporting**, implemented as a scheduled batch pipeline with agent-based data retrieval, LLM-driven interpretation, a Python integration layer and a self-contained browser dashboard.

18. Evidence & Audit Notes

This document is based on the supplied project instructions/context, the supplied live **unified_dispatcher.py**, the supplied **engine_live.js**, the supplied **Service_Insight_Agent_v1_0.json**, and the supplied Claude Project screenshot.

The supplied artifacts support the architecture, workflow structure, configuration, schedules, data contracts and technical decisions described above. Full agent-by-agent SQL/table logic for every KPI, complete ShopZop/VZY source code, deployment scripts and all n8n workflow JSONs were not included in the supplied evidence, so those areas are intentionally not presented as fully audited.

The next audit stage should capture each KPI workflow JSON and the remaining frontend/template files, then create an agent dependency matrix covering input → MCP → table/query logic → transformation → output schema → dispatcher → visualization → insight.

END OF CURRENT-STATE REVERSE ENGINEERING PACK