Requirement

Select ONE high-value asset type. Target a specific model of a generator, compressor, or pump that is common in the Houston O&G or power generation sector.

Vertical

- Power Generation & Data Centers
- Oil & Gas
- Manufacturing
- Commercial & Industrial HVAC

smart meters, SCADA systems, grid sensors

1.0 Core Capabilities Specification: The Guard - A Two-Tiered Anomaly Surveillance System

---

1.1 Introduction

This document specifies the requirements for an Anomaly Surveillance System. The system's primary purpose is to process high-volume telemetry data from a fleet of assets to proactively identify, validate, and contextualize subtle anomalies. These anomalies may be indicative of incipient failures or significant operational inefficiencies.

This document specifies the design for a GenAI-Powered Predictive Maintenance Agent System. The system aims to proactively identify subtle operational anomalies in industrial equipment (e.g., generators), investigate their potential causes and significance using Artificial Intelligence (AI) and Machine Learning (ML), and provide actionable insights to maintenance and operations teams. This system comprises two main phases: an initial screening phase to identify anomaly candidates and a GenAI-driven investigation phase to analyze these candidates.

Business Problem

Traditional predictive maintenance systems often rely on predefined rules and thresholds, which may miss subtle deviations or complex patterns that are precursors to equipment failure. Investigating these subtle issues requires significant manual effort from experienced engineers to consult manuals, historical data, and apply their expertise. This leads to delayed responses, increased risk of unplanned downtime, and inefficient use of expert resources.

System Objectives

Proactive Issue Identification: Augment rule-based alerting by identifying and investigating subtle anomalies that may indicate incipient failures.

Accelerated Diagnosis: Reduce the Mean Time To Diagnose (MTTD) by automating the gathering and analysis of contextual information, operational manuals, and historical data.

Enhanced Decision Support: Provide clear, concise, and actionable assessments, including hypothesized root causes, risk levels, and recommended actions, to operations and maintenance personnel.

Knowledge Leverage: Effectively utilize existing knowledge embedded in operational manuals, maintenance logs, and expert best practices.

Scalability & Efficiency: Design a system capable of handling telemetry from a large fleet of devices and efficiently investigating numerous anomaly candidates.

Continuous Improvement: Incorporate feedback loops to enable the system (including ML models and LLM interactions) to learn and improve over time.

2.0 Data Ingestion & Pre-processing

The system must provide capabilities to prepare raw time-series data for analysis.

- **2.1 Data Resampling:** The system shall be able to standardize telemetry data to a consistent frequency (e.g., aggregate/downsample 1-second data to 1-minute averages).

- **2.2 Data Imputation:** The system shall provide configurable methods for handling missing data points (e.g., forward-fill, linear interpolation).

3.0 System Overview

"The Guard" is a logical system designed to act as a primary filter for high-volume telemetry data. It operates in two sequential tiers to efficiently identify, validate, and contextualize subtle anomalies that may indicate incipient failures or operational inefficiencies.

The system's purpose is to transform a noisy, high-volume data stream into a high-quality, low-volume stream of actionable "confirmed anomaly" events, which serve as the input for downstream investigation systems (e.g., a GenAI Agent System).

- **3.1 Tier 1 (Wide-Net Filter):** Performs rapid, broad, statistical-based screening on the entire data stream to identify potential anomalies based on a defined rule set. Its goal is to significantly reduce data volume while capturing all potentially significant signals.

- **3.2 Tier 2 (Deep Analysis & Confirmation):** Takes the filtered candidates from Tier 1 and applies more sophisticated analytical and machine learning models to confirm if they are true anomalies, adding context and confidence before output.

4.0 Core Capabilities

4.1 Tier 1: Wide-Net Statistical Filtering Capabilities

The system must support the definition and execution of the following logical patterns to screen the incoming telemetry stream and identify "potential anomaly candidates":

- **4.1.1 Simple Threshold Violations:** Identify values that cross a pre-defined static threshold, representing a warning level rather than a critical alarm.

- **4.1.2 Deviation from Dynamic Baselines:** Identify when a sensor value deviates significantly from its own recent, dynamically calculated normal behavior (e.g., rolling average), especially when contextualized with other metrics (e.g., stable engine load).

- **4.1.3 Volatility Shift Detection:** Identify when a metric's stability changes significantly, by detecting a sudden increase or decrease in its statistical variance or standard deviation over a defined time window.

- **4.1.4 Gradual Drift Detection:** Identify slow, persistent, directional changes over an extended period, indicating potential component wear. This includes detecting both a consistent trend and a value's approach toward a known operational limit.

- **4.1.5 Stagnant Value Detection ("Stuck Sensor"):** Identify when a sensor's value remains static for an unusual length of time, particularly when correlated metrics indicate the system's operational state is changing.

- **4.1.6 Co-occurrence of Minor Events:** Identify when a combination of individually minor, non-critical events or alerts occurs within a specified time frame, suggesting a more complex, correlated issue.

- **4.1.7 Deviation from Peer Group (Fleet Analysis):** Identify a device that is behaving as a statistical outlier compared to its peers (e.g., identical models operating under similar load conditions).

4.2 Tier 2: Deep Analysis & Confirmation Capabilities

For each "potential anomaly candidate" identified by Tier 1, the system must be able to perform a deeper analysis to confirm its significance. It must support the following confirmation methods:

- **4.2.1 Time-Series Decomposition Analysis:** The ability to break down a metric's time-series into its trend, seasonality, and residual components. A true anomaly is confirmed if there is a significant spike or dip in the residual component after accounting for normal cyclical patterns.

- **4.2.2 Predictive Outlier Detection (Forecasting):** The ability to build a predictive model based on a metric's recent history, forecast its expected value and confidence interval, and confirm an anomaly if the actual observed value falls outside this predicted range.

- **4.2.3 Multivariate Anomaly Detection:** The ability to model the normal relationships between multiple interacting metrics. An anomaly is confirmed when this relationship is broken, even if each individual metric is within its normal range. This includes comparing a device's multi-metric performance against a cluster of its peers.

- **4.2.4 Pluggable Custom Model Execution:** The architecture must support the integration and execution of custom-trained, external machine learning models (e.g., autoencoders, isolation forests, supervised classifiers) as a final, definitive confirmation step.

4.3 State, Resilience, & Configuration Capabilities

- **4.3.1 Anomaly State Management:** The system must maintain the state of "active" anomalies. It shall not generate a new confirmed anomaly output for an issue that is already in an active state, but it must track when the condition was last seen. The system must also have a mechanism to detect when an active anomaly condition has resolved.

- **4.3.2 Mass Event Suppression ("Thundering Herd" Protection):** The system must be able to detect when a large percentage of the device fleet reports potential anomalies simultaneously. In such an event, it must suppress individual anomaly confirmations and instead issue a single, "fleet-wide" event notification.

- **4.3.3 Dynamic Configuration:** All analytical parameters—including thresholds, time window sizes, statistical factors, and model versions to be used—must be externally configurable on a per-device-model and/or per-metric basis. These configurations must be updatable without requiring a system redeployment.

- **4.3.4 "Learning Mode" for New Assets:** The system must support a mechanism to temporarily exclude specific devices from anomaly detection. This allows new or recently serviced assets to establish a stable operational baseline before being subjected to surveillance, preventing an initial flood of false positives.

5.0 Options for Analytics & Machine Learning Implementation

The following are the categories of techniques that should be considered for implementing the system's analytical capabilities. The final choice of implementation will depend on data characteristics, performance requirements, and desired accuracy.

5.1 Options for Feature Engineering

The process of transforming raw data into inputs for ML models should consider the following types of features:

- **5.1.1 Statistical Features:** Rolling calculations over various time windows (e.g., mean, median, standard deviation, min/max, skewness, kurtosis).
- **5.1.2 Frequency Domain Features:** For relevant sensor data (e.g., vibration, acoustics), apply transformations like the Fast Fourier Transform (FFT) to extract features related to specific frequencies that may indicate wear or imbalance.
- **5.1.3 Time-Based Features:** Cyclical features such as hour of the day, day of the week, or month, and operational features like uptime or time since last maintenance.
- **5.1.4 Delta & Rate-of-Change Features:** The difference or rate of change in sensor readings over time.

5.2 Options for Machine Learning Model Training

The following classes of models should be evaluated for the anomaly confirmation (Tier 2) and peer-group analysis capabilities.

5.2.1 Supervised Learning (for predicting known failure types):

- Requires: A well-defined set of labeled data where "failure" and "normal" states are clearly marked.

- Model Frameworks:
    - Gradient Boosted Trees (e.g., XGBoost, LightGBM) - Typically high-performing on structured, tabular data.
    - Ensemble Methods (e.g., Random Forest) - Robust and less prone to overfitting.
    - Deep Learning (e.g., Neural Networks) - Suitable for capturing highly complex, non-linear patterns if sufficient data is available.

5.2.2 Unsupervised Learning (for detecting novel or "weird" behavior):

- Requires: A large amount of data representing normal operation; does not require failure labels.
- Model Frameworks:
    - Isolation-Based (e.g., Isolation Forest) - Effective at identifying anomalies as points that are "easy to separate" from the dense normal data.
    - Reconstruction-Based (e.g., Autoencoders - a type of neural network) - Models are trained to reconstruct "normal" data. Anomalies are identified by high reconstruction error.
    - Clustering-Based (e.g., DBSCAN) - Anomalies are identified as points that do not belong to any cluster of normal data.

5.2.3 Time-Series Forecasting Models (for predictive outlier detection):

- Requires: Sufficient historical data for a given metric to establish patterns.
- Model Frameworks:
    - Statistical Models (e.g., ARIMA, Prophet)
    - Deep Learning Models (e.g., LSTMs, GRUs)

6.0 Areas Requiring Further Analysis & Definition

Before development, the following areas must be investigated and clearly defined to ensure the success of the system.

- **6.1 Exploratory Data Analysis (EDA):** A thorough analysis of the historical telemetry data is required to validate the assumptions that the patterns and anomalies described in this document actually exist and are detectable.

- **6.2 Formal Labeling Strategy:** For any supervised learning approach, a rigorous and unambiguous definition of a "failure event" is critical. This includes:

    - What constitutes a failure? (e.g., a specific fault code, an unscheduled maintenance event).

    - What is the prediction window? (e.g., are we predicting a failure within the next 7 days, 30 days?).

- **6.3 Fleet Homogeneity Assessment:** The "Deviation from Peer Group" capability relies on the assumption that assets in a group are truly comparable. Analysis is needed to confirm if operational context (e.g., environment, usage patterns) must be used to create more accurate peer groups.

- **6.4 Success Metrics Definition:** Clear business and technical metrics must be defined to evaluate the system's performance. These should include:

    - Technical Metrics: Precision, Recall, F1-Score for anomaly detection.

    - Business Metrics: Reduction in false positive alerts, number of successfully pre-empted failures, reduction in maintenance costs.

- **6.5 Model Explainability (XAI):** A strategy must be defined for how the system will explain why it has flagged an anomaly, especially for complex machine learning models. This is crucial for operator trust and effective root cause analysis.

- **6.6 Feedback Loop Mechanism:** A process must be designed for how insights from downstream investigations (i.e., whether a "confirmed anomaly" was a true failure, a false positive, or a known condition) will be fed back into the system to retrain models and refine rules.

Review and merge below sections

The AI Agent's Core Capabilities

1. Proactive Anomaly Detection (The "Eyes and Ears")

- **Unsupervised Learning:** Using models like Isolation Forests or Autoencoders, it learns the "normal" multi-dimensional fingerprint of a healthy generator. It can then flag any data point that doesn't conform to this learned normal, even if no single value has crossed a static threshold. This is crucial for catching subtle, "unknown unknown" problems.
- **Multivariate Correlation:** The agent understands that while voltage and current might be individually within limits, their *relationship* might be anomalous, indicating an efficiency loss or an impending component failure.
- **Time-Series Forecasting:** It predicts the next few minutes of a sensor's readings. If the actual reading deviates significantly from the prediction, it flags an anomaly.
- **Types of Models:**
    - **Time-series anomaly detection models** (e.g., ARIMA, Prophet-based, autoencoders, LSTMs) trained on historical data to learn "normal" patterns for specific `MeasurementKey`s or `DeviceID`s.
    - **Clustering models** (e.g., K-Means, DBSCAN) to identify devices behaving differently from their peers.

2. Automated Investigation & Root Cause Analysis (The "Brain")

When an anomaly is detected, the agent's reasoning engine kicks in. This is where it mimics an experienced engineer.

- **Automated Data Correlation:** The agent automatically queries the ADX database for related data streams around the time of the anomaly.

    - *Agent's "Thought":* "I see a subtle vibration spike on `generator-Ig1500-40201`. I will now pull the temperature, oil pressure, and load data for the 10 minutes surrounding this event."
- **Historical Pattern Matching:** It compares the current anomalous pattern against a library of historical failure signatures stored in ADLS.
    - *Agent's "Thought":* "This specific high-frequency vibration pattern, combined with a slow temperature increase, matches the signature of the `coolant-pump-bearing-failure` event from Generator-C six months ago with 87% confidence."
- **Knowledge Base Integration (RAG - Retrieval-Augmented Generation):** This is the most powerful part. You digitize your technical manuals, maintenance logs, and engineering schematics and place them in a searchable knowledge base. The agent queries this knowledge base using natural language.
    - *Agent's "Thought":* "The anomaly triggered Modbus `MeasurementKey inverter_fault_code` with a value of `53`. I will now query the knowledge base: 'What does fault code 53 mean for a generator model Ig1500 according to the technical service manual?'"

3. Actionable Insight Generation (The "Voice")

The agent's final step is to translate its complex findings into a simple, actionable insight for the human teams. It doesn't just present data; it presents a conclusion.

- **Plain-Language Summary:** It uses a Large Language Model (LLM) to generate a concise summary of the issue.
- **Severity Assessment:** It assigns a priority level (e.g., P1-Urgent, P3-Monitor).
- **Recommended Actions:** Based on the manual and past successful repairs, it suggests concrete steps.

Example of an AI Agent's Output (e.g., a ticket in ServiceNow or a Teams message):

Subject: [P2-Advisory] Proactive Alert for Generator-Ig1500-40201

Summary: A subtle, high-frequency vibration pattern has been detected, starting at 02:15 UTC. While all primary metrics are within limits, this pattern strongly correlates (87% match) with a known precursor to coolant pump bearing failure.

Investigation:

- **Anomaly:** Sustained 0.8g vibration above the learned baseline.
- **Correlated Data:** A slow 2°C rise in coolant temperature was observed over the same period.
- **Knowledge Base:** The service manual for the Ig1500 (Doc #7B, page 42) indicates these are early signs of bearing wear. A similar event on Generator-C led to failure within 14 days.

Recommended Action:

1. Schedule a physical inspection of the coolant pump on Generator-Ig1500-40201 within the next 3-5 days.
2. Prioritize checking the pump bearing for signs of spalling or excessive play.
3. **Required Parts:** Coolant Pump Assembly (Part #78-2B-9)

Benefits of the AI Agent Approach

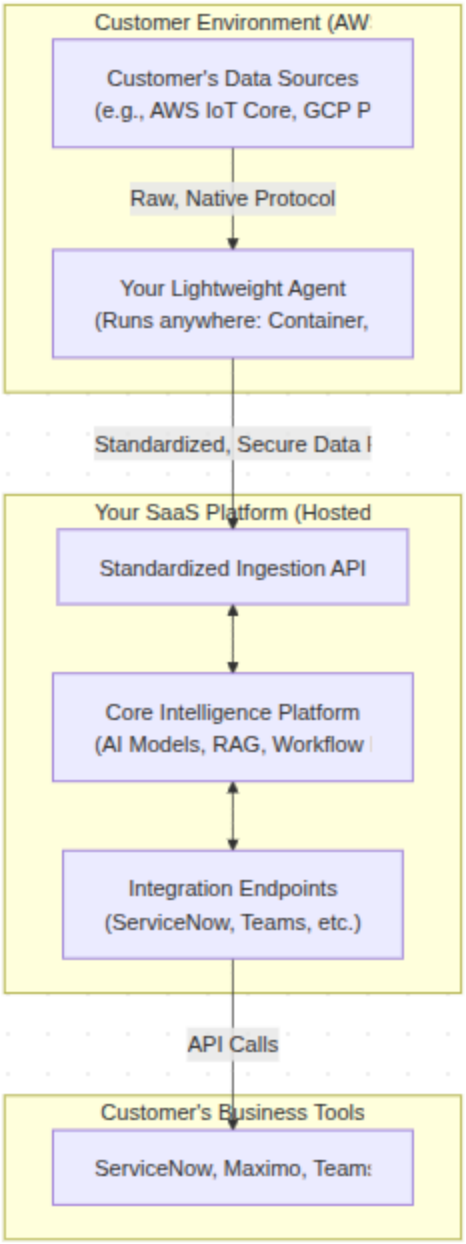
1. **Frees Up Expert Resources:** Your best engineers are no longer consumed by initial data investigation. They are presented with a pre-investigated problem, allowing them to focus on complex validation and strategic repairs.
2. **Reduces Mean Time to Resolution (MTTR):** The investigation process is reduced from hours or days to seconds.
3. **Captures & Scales Institutional Knowledge:** The expertise of your most senior engineer, once trapped in their head, is now encoded into the Knowledge Base and the agent's logic, available 24/7 to the entire team.
4. **Prevents Unplanned Downtime:** By catching subtle precursors to failure, you move from reactive/predictive to truly *prescriptive* maintenance.
5. **Creates a Learning Loop:** When a technician confirms the fix, that feedback can be used to retrain the models, making the agent smarter and more accurate over time.

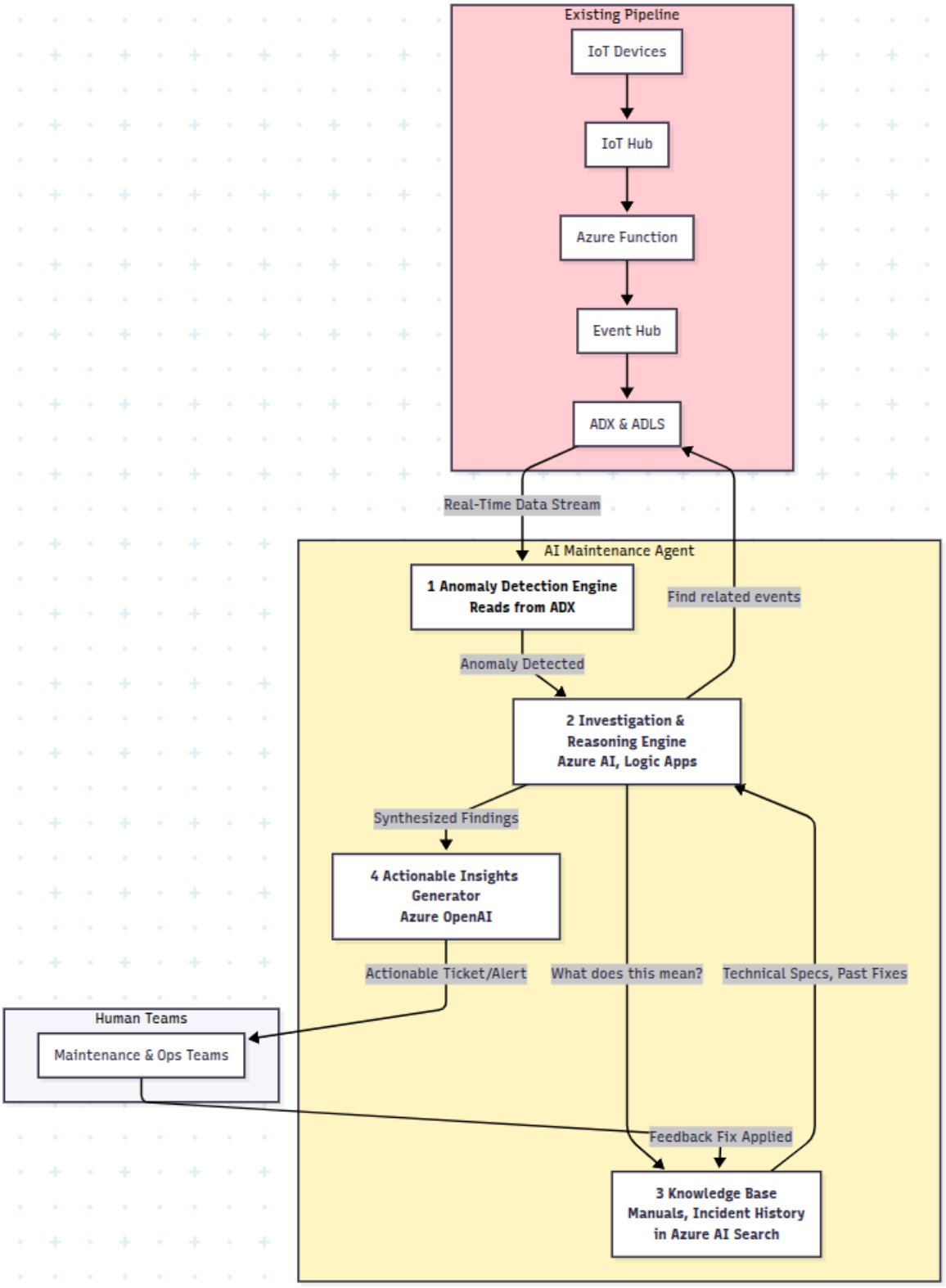
Highlevel Architecture

Highlevel Architecture

The Core Strategy: Decouple Product's Brain from the Customer's Environment

1. **The Data Collector/Agent:** A lightweight component that lives in the customer's environment.
2. **The Core Intelligence Platform:** Your SaaS application (the "brain") that hosts the AI models, knowledge base, and reasoning engine. This is where your IP lives.


Customer Environment (AW
Customer's Data Sources
(e.g., AWS IoT Core, GCP P
Raw, Native Protocol
Your Lightweight Agent
(Runs anywhere: Container,
Standardized, Secure Data I
Your SaaS Platform (Hosted
Standardized Ingestion API
Core Intelligence Platform
(AI Models, RAG, Workflow
Integration Endpoints
(ServiceNow, Teams, etc.)
API Calls
Customer's Business Tools
ServiceNow, Maximo, Team


Existing Pipeline
IoT Devices
IoT Hub
Azure Function
Event Hub
ADX & ADLS
Real-Time Data Stream
AI Maintenance Agent
1 Anomaly Detection Engine
Reads from ADX
Find related events
Anomaly Detected
2 Investigation &
Reasoning Engine
Azure AI, Logic Apps
Synthesized Findings
4 Actionable Insights
Generator
Azure OpenAI
Actionable Ticket/Alert
What does this mean?
Technical Specs, Past Fixes
Human Teams
Maintenance & Ops Teams
Feedback Fix Applied
3 Knowledge Base
Manuals, Incident History
in Azure AI Search

Data Source

Data Source

- CCMS
- ERP
- SCADA
- Historical Database
- Scheduling System
- Ordering System

Pre-filtering

Baseline & Statistical Monitoring(Pre-filtering)

- **What happens:** As data flows into the time-series datastore, automated systems (could be built-in features of the datastore, or separate stream processing jobs like Apache Flink, Kinesis Data Analytics, Azure Stream Analytics) continuously monitor key measurements for each device (or device type).
- **How it works:** These systems calculate statistical baselines (e.g., moving averages, standard deviations) for "normal" behavior over different time windows (e.g., last hour, last 24 hours, same time last week). They look for:
    - **Drifts:** A sensor value slowly creeping up or down over time, even if still within absolute defined limits.
    - **Increased Volatility:** A sensor value that used to be stable starts fluctuating more wildly.
    - **Correlated Deviations (Simple):** Maybe a slight increase in temperature *consistently* appears with a slight decrease in efficiency, even if both are "within spec."
- **Output:** This stage doesn't say "it's an anomaly." It says, "This specific generator (`DeviceId`) and these specific measurements (`MeasurementKey`s) are behaving *unusually* compared to their recent history or the fleet average. They are *candidates* for deeper investigation." This significantly narrows down the data.
- **Load Management:** This is computationally cheaper than full LLM analysis. It acts as a high-throughput filter.

Rules/Types of Screening Logic

- **Threshold Violations (Simple but still useful pre-filter):**
- **Rate of Change / Drifts:** Comparing current values to a short-term moving average:
- **Deviation from Statistical Norms**
- **Simple Correlated Deviations**
- **Counting Occurrences:**

Anomaly Type	Concept	SAQL Example Idea

Deviation from Short-Term Dynamic Baselines	A sensor value is normally stable or follows a predictable pattern based on load/time of day. A sudden deviation, even if small, is interesting.	Calculate a 5-minute rolling average for engine_temperature per DeviceId. Flag if the current engine_temperature is > 10% above this rolling average for more than 60 consecutive seconds, *while engine_load has remained relatively constant*. (The load condition helps filter out normal temp changes due to load changes).
Increased Volatility/Noise	A sensor reading that used to be very smooth starts fluctuating erratically. This can indicate sensor issues or mechanical instability.	Calculate the standard deviation of output_voltage over a 1-minute tumbling window. Flag if this standard deviation suddenly jumps significantly compared to its average standard deviation over the last hour.
Co-occurrence of Minor Non-Critical Events	Individually, these events are minor. Together, they might paint a picture.	Flag if DeviceId reports filter_pressure_warning (a non-critical informational event) AND minor_vibration_alert within the same 10-minute window.
Stagnant Values (When Change is Expected)	Some sensors should change with operational state. If they get "stuck," it's an issue.	If fuel_flow_rate remains exactly the same value for > 5 minutes while engine_load has changed by > 20%, flag it.
Gradual Drifts Approaching Warning Thresholds	A value is slowly creeping towards a known warning limit but hasn't crossed it yet.	If exhaust_temperature is within 5% of its documented pre-alarm warning threshold AND has shown a consistent upward trend over the last 30 minutes, flag it.
Comparison to "Twin" or "Fleet Average" (More Advanced)	If a generator starts behaving differently from its identical peers under similar conditions.	If generator_A_efficiency drops by 5% compared to the average efficiency of other generators of the same model operating at a similar load percentage in the last hour, flag it. This is more complex in ASA but powerful.

Agents names and functions

Agents names and functions

Agents functions

Agents functions

1. **Orchestrator Agent (The "Project Manager")**
    a. To manage the end-to-end investigation workflow for a "subtle anomaly candidate" received from Phase 1 (ASA). It directs the other agents and sequences their tasks.
2. **Context Agent (The "Field Investigator & Data Analyst")**
    a. To gather detailed, real-time and recent historical operational data relevant to the anomaly candidate.
3. **Knowledge Agent (The "Librarian & Historian")**
    a. To retrieve relevant information from structured and unstructured knowledge sources (manuals, historical maintenance logs, best practices).
4. **Reasoning Agent (The "Chief Diagnostician & Strategist")**
    a. To synthesize all gathered information (candidate details, operational context, knowledge snippets) and perform the core diagnostic reasoning to assess the anomaly's significance and formulate hypotheses. This is where the primary LLM interaction for deep analysis occurs.
5. **Action & Reporting Agent (The "Communicator & Dispatcher")**
    a. To communicate the findings and recommendations of the investigation to relevant stakeholders or systems.

1. Proactive Anomaly Detection (The "Eyes and Ears")
2. Automated Investigation & Root Cause Analysis (The "Brain")
3. Actionable Insight Generation (The "Voice")

- **Agent 1: Ingestion**
    - Manages workflow, delegates.
- **Agent 2: Anomaly Detection Agent.**
- **Agent 3: Orchestrator Agent**
- **Agent 4: Chain-of-Thought (CoT) Diagnostician Agent.**
    - Synthesizes, diagnoses, hypothesizes (core LLM work).
- **Agent 5: RAG-Powered Researcher Agent.**
    - Retrieves from manuals, historical logs (RAG).
- **Agent 6: Summary & Action Generator Agent.**
    - Communicates findings, recommends actions.

1. **Our Orchestrator Agent:**
    - **Adopt:** Priority-based routing (which sub-functions/agents it calls and with what urgency), resource management considerations (awareness of sub-function load/availability), and the specific performance/monitoring metrics.
    - **Decision:** It can still delegate initial context enrichment to the Context Agent for clarity.
2. **Our Context Agent:**
    - **Adopt:** The comprehensive list of data to retrieve (equipment info, detailed historical performance, maintenance context, fleet comparison, environmental context) as its target data gathering scope. It will use ADX and potentially other structured data APIs.
3. **Our Knowledge Agent:**
    - **Reinforced Role:** Its role in RAG (manuals, known issues, historical unstructured logs) is validated. It would work closely with the Context Agent's structured data.
4. **Our Reasoning Agent:**
    - **Internalize "Analysis Agent" functions:** It can perform statistical assessment, pattern recognition, and initial risk assessment as a first step (perhaps through specific LLM prompts or by calling helper functions/models) *before* diving into the deeper root cause diagnosis using RAG outputs from the Knowledge Agent.
    - **Adopt:** The detailed diagnostic outputs, including hypotheses, confidence, investigation steps, and even "Detailed Procedures."
5. **Our Action & Reporting Agent:**
    - **Adopt:** The detailed scope of action planning (prioritization, resource planning, work order considerations, scheduling inputs, risk management for the plan). It can evolve from just "reporting" to more active "planning assistance."

Orchestrator Agent

Orchestrator Agent

. Orchestrator Agent

- **Purpose:** Manages the entire investigation lifecycle, coordinates other agents, and ensures timely and prioritized processing.
- **Functionalities:**
    1. **Initiate Investigation:**
        - Receives `Subtle Anomaly Candidate` message.
        - Logs initiation, assigns an `InvestigationID`.
        - Determines/confirms investigation `Priority` (if not already set or needs re-evaluation).
        - Based on `Priority`, defines the scope of investigation (e.g., depth of historical data, which knowledge sources are mandatory).
    2. **Delegate to Context Agent:**
        - Sends `DeviceID`, `FlaggedMeasurements`, `TimestampDetected`, and required `ContextDepth` (based on priority) to the Context Agent.
        - Awaits `OperationalContextPackage`.
    3. **Delegate to Knowledge Agent:**
        - Sends `DeviceID`, `DeviceModel` (if known or obtainable via Context Agent), `FlaggedMeasurements`, a summary of observed behavior (from candidate + context), and specific areas of interest (e.g., "causes for X," "similar past Y") to the Knowledge Agent.
        - Awaits `KnowledgeSnippetsPackage`.
    4. **Delegate to Reasoning Agent:**
        - Compiles: `Subtle Anomaly Candidate`, `OperationalContextPackage`, `KnowledgeSnippetsPackage`.
        - Sends this comprehensive package to the Reasoning Agent.
        - Awaits `DiagnosticAssessmentPackage`.
    5. **Delegate to Action & Reporting Agent:**
        - Sends `DiagnosticAssessmentPackage` (containing diagnosis, recommendations, confidence, urgency) to the Action & Reporting Agent.
        - May also pass original candidate details and key context for reporting.
    6. **Monitor & Manage Workflow:**
        - Tracks timeouts for each sub-agent's task based on priority.

    - Handles errors from sub-agents (e.g., retry logic, fallback strategies if a data source is unavailable).
    - Manages investigation queues if candidate arrival rate exceeds processing capacity, prioritizing based on `Priority`.
7. **Conclude Investigation:**
    - Logs completion, outcome, and key metrics (e.g., time to diagnose).

AI Agents_Gemini_Deepsearch

Core Architecture and Unique Value Proposition (UVP)

The solution is conceived as an intelligent, automated system that mimics and accelerates the workflow of a human reliability expert. It is composed of four distinct but interconnected AI agents, each with a specialized function in the "detect-to-correct" maintenance lifecycle.

- **Agent 1: Ingestion & Anomaly Detection Agent.** This agent serves as the system's sensory input. It establishes secure connections to the client's existing operational technology (OT) infrastructure, including SCADA systems, data historians, and IoT sensor gateways.[53] It ingests and normalizes high-frequency telemetry data streams—such as vibration, temperature, pressure, and acoustic signals. Using a suite of proven machine learning models (e.g., time-series analysis, anomaly detection), this agent continuously monitors asset behavior against learned baselines to detect statistically significant deviations that indicate a potential failure. The detection of an anomaly is the trigger that activates the rest of the system. This capability, while essential, is considered table stakes in the modern PdM market.[55]
- **Agent 2: Chain-of-Thought (CoT) Diagnostician Agent.** This is the cognitive core of the system and the primary source of differentiation. Upon receiving an alert from the Anomaly Detection Agent, the CoT Diagnostician initiates a structured reasoning process.[7] It does not simply flag the anomaly; it begins to investigate it by forming and testing hypotheses. For example, if alerted to "high-frequency vibration and a temperature spike on Pump-123," the agent's internal monologue would follow a logical path: "Initial alert on Pump-123. High vibration and heat are common symptoms of bearing failure, shaft misalignment, or lubrication issues.
    - Step 1: Query internal maintenance database for recent work on Pump-123.
    - Step 2: Query operational data for associated pressure drops, which could indicate cavitation.
    - Step 3: Task the RAG agent to retrieve the specific troubleshooting section for 'vibration' from the Pump-123 OEM manual." This structured, step-by-step diagnostic process mirrors how a human expert would approach the problem.
- **Agent 3: RAG-Powered Researcher Agent.** This agent acts as the diagnostician's research assistant. It takes the queries generated by the CoT agent and executes them against a vectorized knowledge base. This knowledge

base is a critical asset, created during client onboarding by ingesting and indexing all relevant documentation:
    - OEM equipment manuals, standard operating procedures (SOPs), historical maintenance work orders, and past incident reports.[5]
    - The RAG agent can retrieve highly specific and contextually relevant information, such as component diagrams, torque specifications, recommended lubricant types, or notes from a technician who solved a similar problem two years prior.[7]
- **Agent 4: Summary & Action Generator Agent.** This final agent acts as the communications interface. It synthesizes the logical reasoning path from the CoT agent and the documentary evidence from the RAG agent into a concise, human-readable report.[5] It generates three key outputs:
    - 1) A plain-language summary for managers explaining the likely root cause and business impact.
    - 2) A detailed diagnostic report for engineers outlining the evidence.
    - 3) A pre-populated, step-by-step work order for technicians, recommending specific actions, required parts, and safety procedures (e.g., Lockout/Tagout).[6]

This architecture supports a clear and powerful **Unique Value Proposition (UVP):** *"We don't just give you alerts; we give you answers. Our AI autonomously investigates equipment failures, providing a complete diagnostic report and an actionable repair plan in minutes, not hours."*

Phases

Phases

End-to-End Workflow: From Raw Data to Actionable Insight

Phase 1: High-Speed Data Collection & Preparation (The Data Factory)

This phase runs continuously, 24/7, and is built for speed and scale.

- **Step 1: Ingestion.** Thousands of compressed messages per second from all your generators stream into **Azure IoT Hub**. IoT Hub acts like a giant, multi-lane post office sorting mail from thousands of sources simultaneously.
- **Step 2: Processing & Structuring.** As messages arrive, they instantly trigger the **Azure Function**. This function is highly efficient. It works on batches of messages (not just one at a time), decompresses them, flattens them into individual records, and sends them to the **Azure Event Hub**. Think of this as the high-speed processing line in the post office, getting every letter ready for the next stage.
- **Step 3: Storing for Analysis. Azure Data Explorer (ADX)** is constantly listening to the Event Hub. It pulls in the structured data in near real-time (within seconds) and organizes it into a massive, time-ordered database table. This is the central library where all generator data is stored, indexed by time, and ready for immediate querying. **ADX is the workhorse that makes this all possible.**

Phase 2: Continuous Surveillance (The "Guard on Duty")

This phase is where the "proactive" part begins. The AI Agent isn't one single thing but a set of automated guards watching the data in ADX.

- **Step 4: The Anomaly Detection Scan.** Every minute (or on a similar schedule), an automated job runs a special query in ADX. This isn't a simple `SELECT *` query. It uses powerful, pre-built ADX functions to analyze the patterns in the latest data.
    - **Guard 1 (The Rule-Checker):** Checks for obvious rule violations (e.g., temperature > 95°C).
    - **Guard 2 (The Pattern-Matcher):** Looks for known bad signatures (e.g., the specific vibration pattern that often precedes a bearing failure).
    - **Guard 3 (The "Odd-One-Out" Detector):** This is the most advanced guard. It uses a machine learning model to find any generator that is behaving differently from its

peers or differently from its own history, even if no rules are broken. It's looking for "subtle weirdness."

- **Step 5: The Trigger.** For 99.9% of the scans, the guards find nothing unusual, and the process simply waits for the next minute. However, if any of the guards detect an anomaly—a rule break, a pattern match, or just "weirdness"—it **triggers an alert**. This alert isn't for a human yet; it's a signal to wake up the next part of the AI Agent.

Phase 3: Automated Investigation (The "Detective")

This is the core of the AI Agent's intelligence. It only runs when triggered by an anomaly, saving massive amounts of computational effort.

- **Step 6: The Investigation Begins.** The alert from the previous step kicks off an **Azure Logic App** (or a similar workflow tool). This Logic App is the detective, and the anomaly is its first clue.
- **Step 7: Gathering Context.** The detective's first action is to go back to the ADX database to gather more evidence. It automatically runs a series of queries:
    - "Show me all sensor data for this specific generator for the 15 minutes before and 5 minutes after the anomaly."
    - "Have any other generators of this model shown a similar pattern in the last 6 months?"
    - "What was the generator's load and operational mode when this happened?"
- **Step 8: Consulting the Knowledge Base.** Now armed with context, the detective queries the **Azure AI Search** knowledge base. This is like the detective going to the library of expert knowledge (manuals, past incident reports). It asks questions in plain language:
    - "What does Modbus fault code `53` mean for a `Ig1500` model?"
    - "What are the documented causes of high-frequency vibration in the coolant pump assembly?"
    - "What was the resolution for `Incident #4321`, which had a similar data signature?"

Phase 4: Synthesis & Recommendation (The "Advisor")

The detective now has all the evidence: the initial anomaly, the surrounding data context, and the relevant information from technical manuals and past incidents.

- **Step 9: Generating the Report.** The Logic App bundles all these findings and sends them to **Azure OpenAI**. It gives the AI a final instruction:

    - "You are an expert generator maintenance engineer. Based on the following data anomaly, contextual sensor readings, and information from the service manual, please provide a concise summary, assess the risk, and recommend a course of action."
-
- **Step 10: Delivering the Actionable Insight.** The OpenAI service generates the clear, human-readable report described in the previous answer. The Logic App then takes this final report and delivers it to the right place:
    - It creates a new high-priority ticket in **ServiceNow** or **Jira**.
    - It posts a detailed alert in a specific **Microsoft Teams channel** for the operations team.
    - It sends an email to the lead maintenance supervisor.

Implementation Details

Implementation Details

Phase 2: Continuous Surveillance (The "Guard on Duty")

Guard 1 (The Rule-Checker):

Checks for obvious rule violations (e.g., temperature > 95°C).

Guard 2 (The Pattern-Matcher):

Looks for known bad signatures (e.g., the specific vibration pattern that often precedes a bearing failure).

Guard 3 (The "Odd-One-Out" Detector):

This is the most advanced guard. It uses a machine learning model to find any generator that is behaving differently from its peers or differently from its own history, even if no rules are broken. It's looking for "subtle weirdness."

Unsupervised learning model

Guard 3" (the "Odd-One-Out" Detector) uses an **unsupervised learning** model. This is a critical distinction from traditional models because you **do not need to label data as "good" or "bad"** beforehand. The model learns what "normal" looks like all by itself.

Here's a step-by-step breakdown of how it works:

Step A: The Training Phase (Done Offline)

This phase happens periodically (e.g., once a week or once a month) using Azure Machine Learning.

1. **Data Extraction:** The system pulls a large, representative sample of historical data from ADX/ADLS. It specifically selects data from periods when the generators were known to be operating normally (e.g., no major alarms, operating under standard load). This might be a few weeks' worth of data across the whole fleet.

2. **Feature Engineering:** It selects the key telemetry streams to learn from—things like `Temperature`, `Voltage`, `Current`, `Vibration`, `OilPressure`. It might also create new features, like the *rate of change* of temperature. This creates a multi-dimensional "fingerprint" for each moment in time.
3. **Model Training:** This data is fed into an unsupervised anomaly detection model, such as:
    - **Isolation Forest:** Imagine the model rapidly asking random questions to partition the data, like "Is the temperature > 40?" and "Is the vibration < 0.2g?". Data points representing normal operation are numerous and clustered together, so they take many questions to isolate. Anomalous data points are rare and different, so they are "isolated" with very few questions. The model learns to identify points that are easy to isolate.
    - **Autoencoder (a type of Neural Network):** This model has two parts. The first part learns to compress the complex multi-dimensional fingerprint of a generator into a much smaller, simplified representation (like a ZIP file). The second part learns to decompress that representation back into the original fingerprint. The model is trained only on *normal* data.
4.

Step B: The Deployment & Inference Phase (Running in Real-Time)

The trained model is now deployed and ready to act as Guard 3.

1. **Real-Time "Fingerprint" Creation:** As new data flows into ADX, the "run every minute" query takes the latest readings for a generator and creates the same multi-dimensional fingerprint used during training.
2. **Anomaly Scoring:** This new fingerprint is fed to the trained model for a "score."
    - **Isolation Forest:** The model checks how many questions it takes to isolate this new data point. If it takes very few, it gets a high anomaly score.
    - **Autoencoder:** The model compresses and then decompresses the new fingerprint. It then measures the **"reconstruction error"**—how different is the decompressed output from the original input? Because the model only learned how to reconstruct *normal* data, it will do a poor job of reconstructing an anomalous fingerprint. A high reconstruction error means a high anomaly score.
3.
4. **The Trigger:** If the anomaly score for any generator's data point crosses a predefined threshold (e.g., "reconstruction error > 0.9"), Guard 3 declares an anomaly and triggers the investigation phase (Phase 3 from the previous workflow).

In simple terms, **Guard 3's model learns the "rhythm" of a healthy machine.** It doesn't know *what* a failure is, but it becomes exceptionally good at detecting when the rhythm is broken, even in a very subtle way that would be invisible to a human looking at raw numbers on a screen.

Test use/data

Test use/data

Select ONE high-value asset type. Target a specific model of a generator, compressor, or pump that is common in the Houston O&G or power generation sector.

Prioritize the `Knowledge Agent` and `Reasoning Agent`. The magic happens when you combine the client's own data with their own manuals to produce a diagnosis.

- **`Knowledge Agent` MVP:** Ingest a limited set of documents for *only the target asset type* (e.g., 1-2 operational manuals, 5-10 key maintenance logs).
- **`Reasoning Agent` MVP:** Focus the LLM prompting on a few critical failure modes for that specific asset.
- **`Action & Reporting Agent` MVP:** Deliver the output as a simple, structured email or PDF report. A fancy UI is not needed for the pilot.