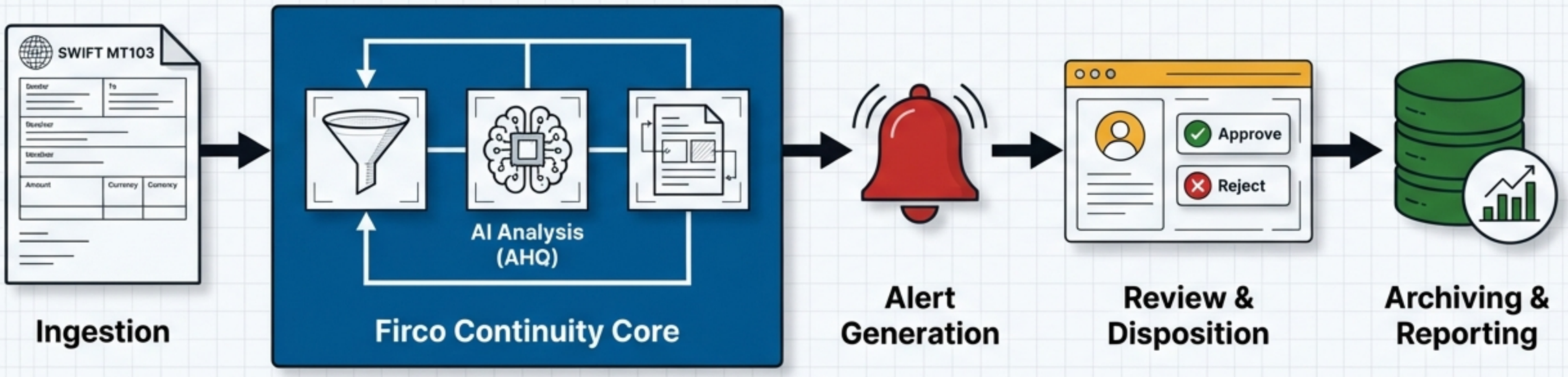
LexisNexis®
RISK SOLUTIONS

Architecting the Firco Continuity Ecosystem

A Technical Blueprint for End-to-End Deployment

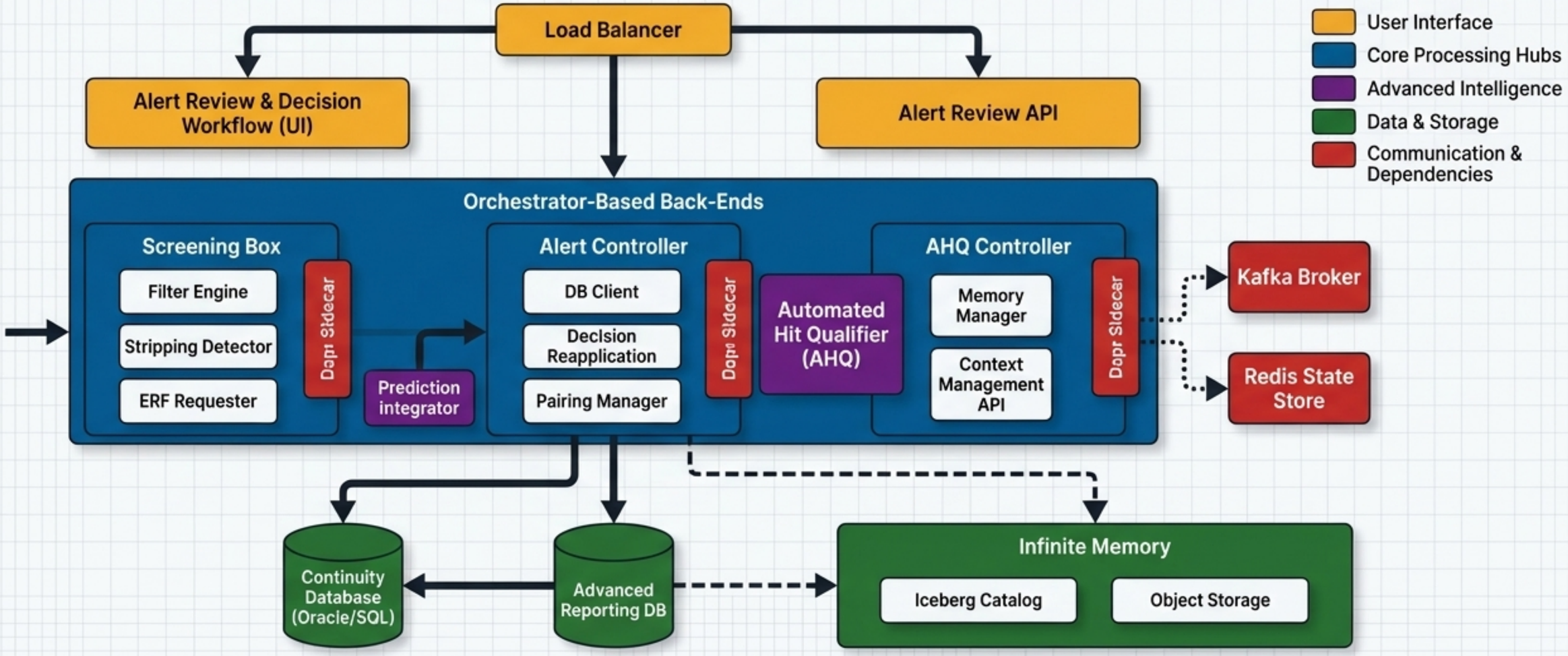
The Mission: From Transaction to Decision in Real-Time

The Firco Continuity ecosystem is designed for one primary mission: to intercept, screen, and disposition financial transactions against sanctions lists and internal policies with high performance and accuracy. The architecture supports the complete lifecycle of an alert, from automated initial filtering to final human review.



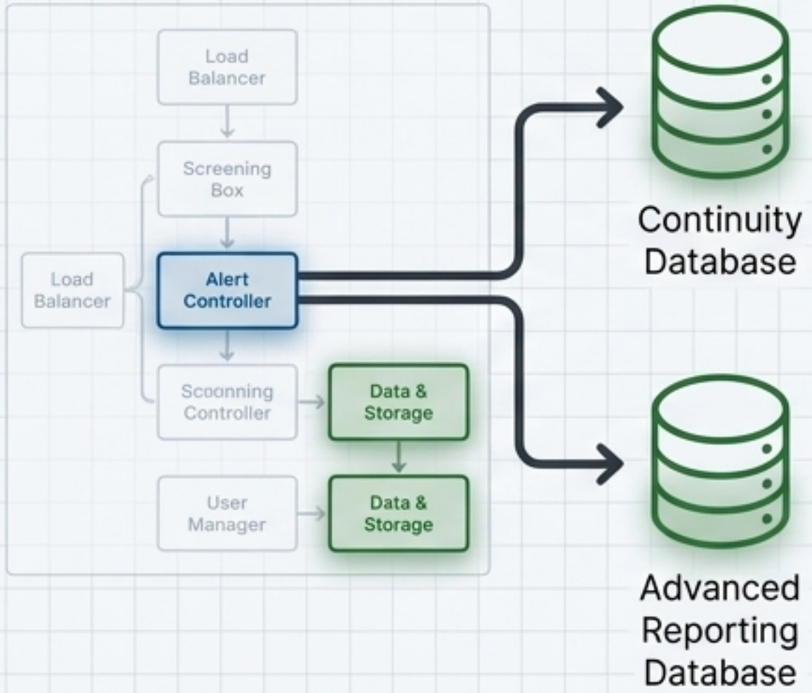
The Master Blueprint: The Firco Continuity Ecosystem

This diagram represents the complete end-to-end architecture. We will build out and detail each colored section, demonstrating how each component contributes to the overall system.



Layer 1: The Foundation - Database Architecture

The system relies on two primary relational databases. All procedures must be performed by a qualified DBA. The Continuity database is the central repository for alerts, user data, and system state.



Oracle

- **Account Privileges:** `CONNECT`, `RESOURCE`
- **Character Set:** `NLS_CHARACTERSET` must be `AL32UTF8`
- **Length Semantics:** `NLS_LENGTH_SEMANTICS` must be `BYTE`
- **Additional Tables:** Run `quartz-schema-oracle.sql` and `batch-schema-oracle.sql` for specific components.

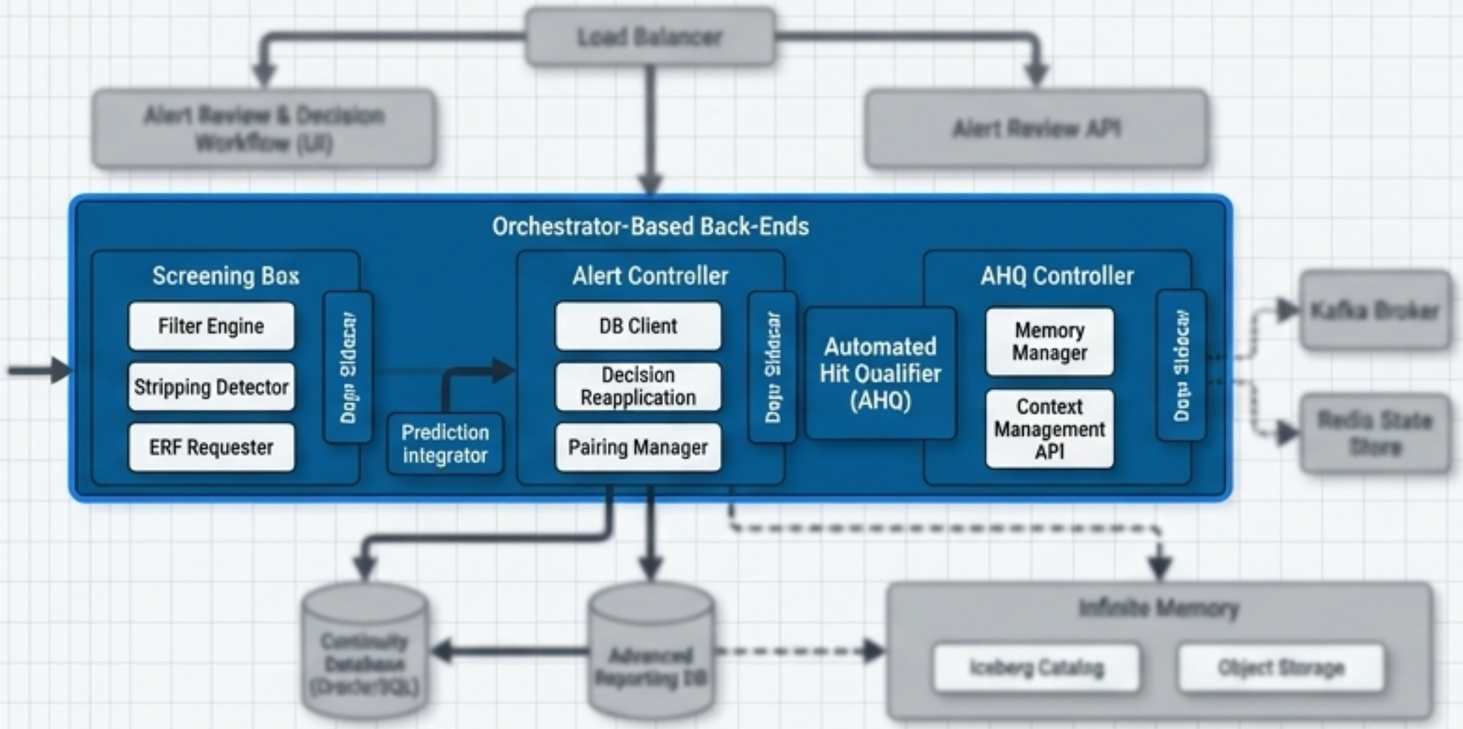
SQL Server

- **Collation:** `SQL_Latin1_General_CP1_CI_AS`
- **Isolation Level:** Read Committed Snapshot Isolation (RCSI) must be enabled.
- **Account Privileges:** User requires schema creation rights.
- **Additional Tables:** Run `quartz-schema-sqlServer.sql` and `batch-schema-sqlServer.sql`.

A license must be registered in the Continuity database after creation before any components can be used.

Layer 2: The Core - Orchestrator Processing Hubs

The system's real-time processing capabilities are handled by three main Orchestrator-based packages. These packages are modular, containing all necessary components for their function. Installation involves extracting a series of versioned sub-packages in a specific order.

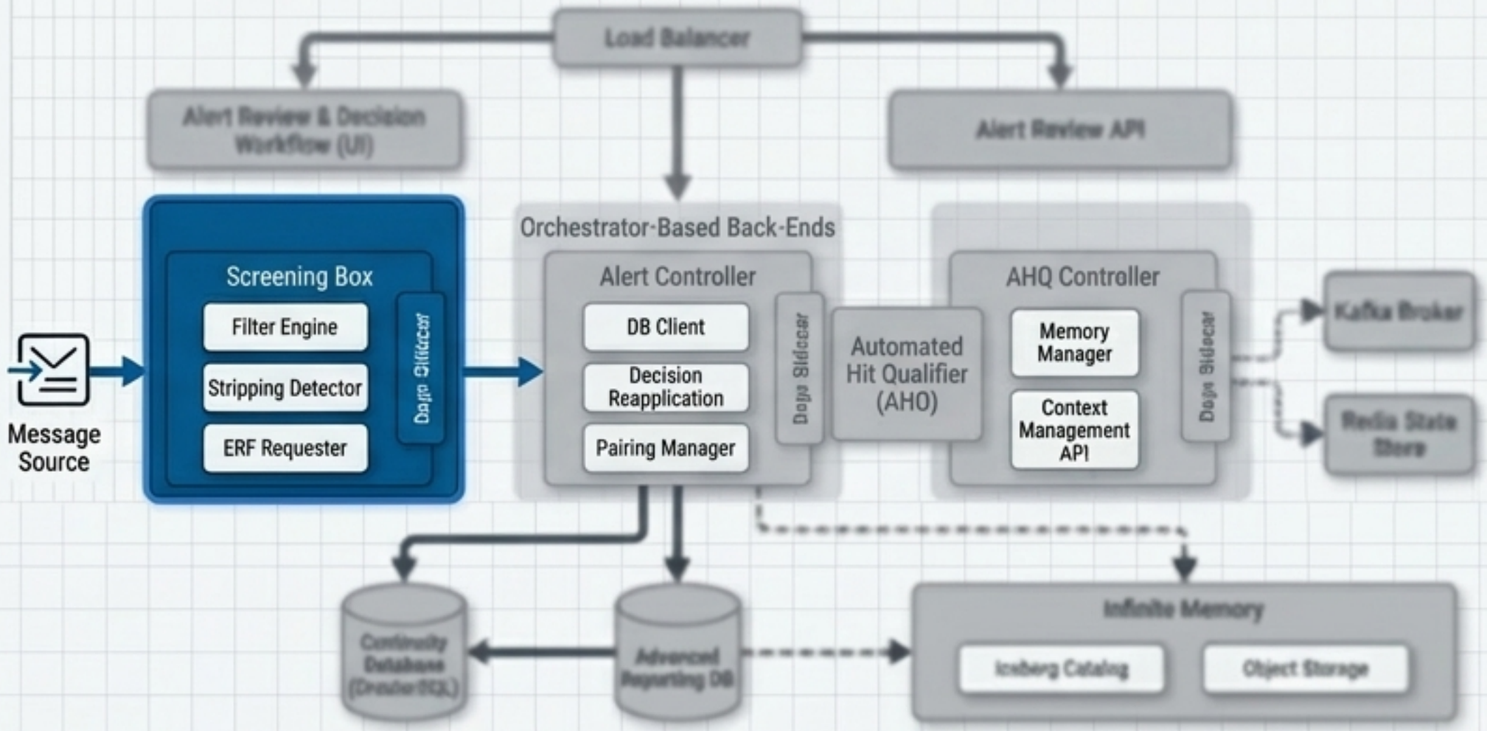


Key Concepts

- **Shared Foundation:** All three packages are built on the 'Orchestrator' core component.
- **Modular Installation:** Components are installed by extracting sub-packages (`**[00-Main]**`, `[01-FilterV5]`, etc.) in a precise sequence. The `**[00-*]** sub-packages are always extracted first.
- **Environment-Driven Configuration:** Each package is configured primarily through `.env` files, allowing for easy separation of configuration from binaries.
- **Flexible Connectivity:** Communication is managed via one of four configurable transport modes: `DIR`, `MQ`, `HTTP`, or `PUBSUB`.

Deep Dive: The Screening Box - Ingestion & Filtering

The Screening Box is the primary entry point for transactions into the Firco Continuity ecosystem. It orchestrates the process of parsing, stripping, filtering, and enriching messages before creating alerts.



Component Breakdown

- **Core Product**: Orchestrator, Filter V5 Engine.
- **Stripping Detector**: Identifies and flags attempts to omit or alter information in payment messages.
- **ERF Requester**: Integrates with the external Entity Resolution Filter for advanced name matching.
- **Automated Hit Qualifier (AHQ)**: Applies machine learning models to automatically disposition simple hits.
- **Prediction Integrator**: Connects to external prediction engines for custom risk scoring.
- **Context Recorder**: Captures detailed message context, mandatory for the Infinite Memory feature.
- **Filter V6**: The next-generation filtering engine.

Key Configuration File: `ScreeningBox/.env` and module-specific files like `.env_ahq`, `.env_stripping`.

Screening Box: Connectivity and Configuration

The Screening Box's interaction with other systems is defined by the selected transport mode, set via the `SCREENING_BOX_TRANSPORT_MODE` variable in `ScreeningBox/.env`. This choice dictates infrastructure requirements.

DIR: File-based queues, simple setup.

MQ: Enterprise-grade messaging via IBM MQ.

HTTP: Synchronous request/reply via REST API.

PUBSUB: Asynchronous, scalable messaging via Dapr.

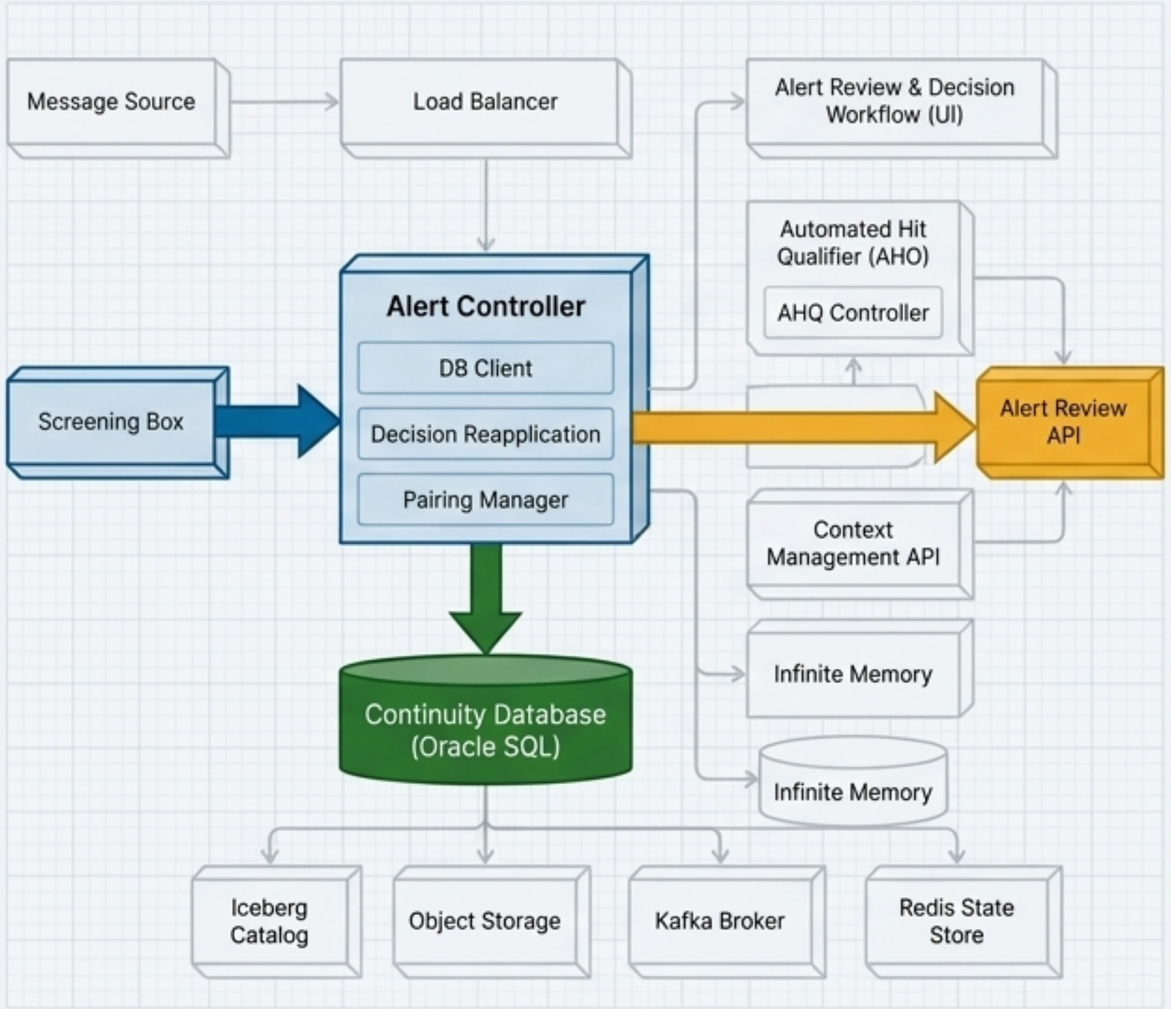
Key Environment Variables

Variable	File	Description
SCREENING_BOX_TRANSPORT_MODE	.env	Sets the primary communication method (DIR, MQ, HTTP, PUBSUB).
SERVER_PORT	.env	Defines the HTTP server port for functional endpoints (e.g., 8080).
MANAGEMENT_SERVER_PORT	.env	Port for admin endpoints (e.g., 8081).
SCREENING_BOX_DAPR_PORT	.env	Dapr sidecar port for PUBSUB mode (e.g., 8888).
FILTERING_REQUESTER_PORT	.env_requester	Port for the core Filter Requester (e.g., 5400).
STRIPPING_ENABLED	.env	Master toggle for the Stripping Detector module.
AHQ_ENABLED	.env	Master toggle for the Automated Hit Qualifier module.

⚠ When deploying multiple instances on one node, all ports must be unique to avoid silent failures, especially with Core Engine back-ends. (Ref: Sec 2, Warning about ports)

Deep Dive: The Alert Controller - Alert Lifecycle Management

The Alert Controller receives alerts from the Screening Box and manages their entire lifecycle. It persists alert data to the database, applies business logic, and provides the API backend for the Alert Review and Decision Workflow front-end.



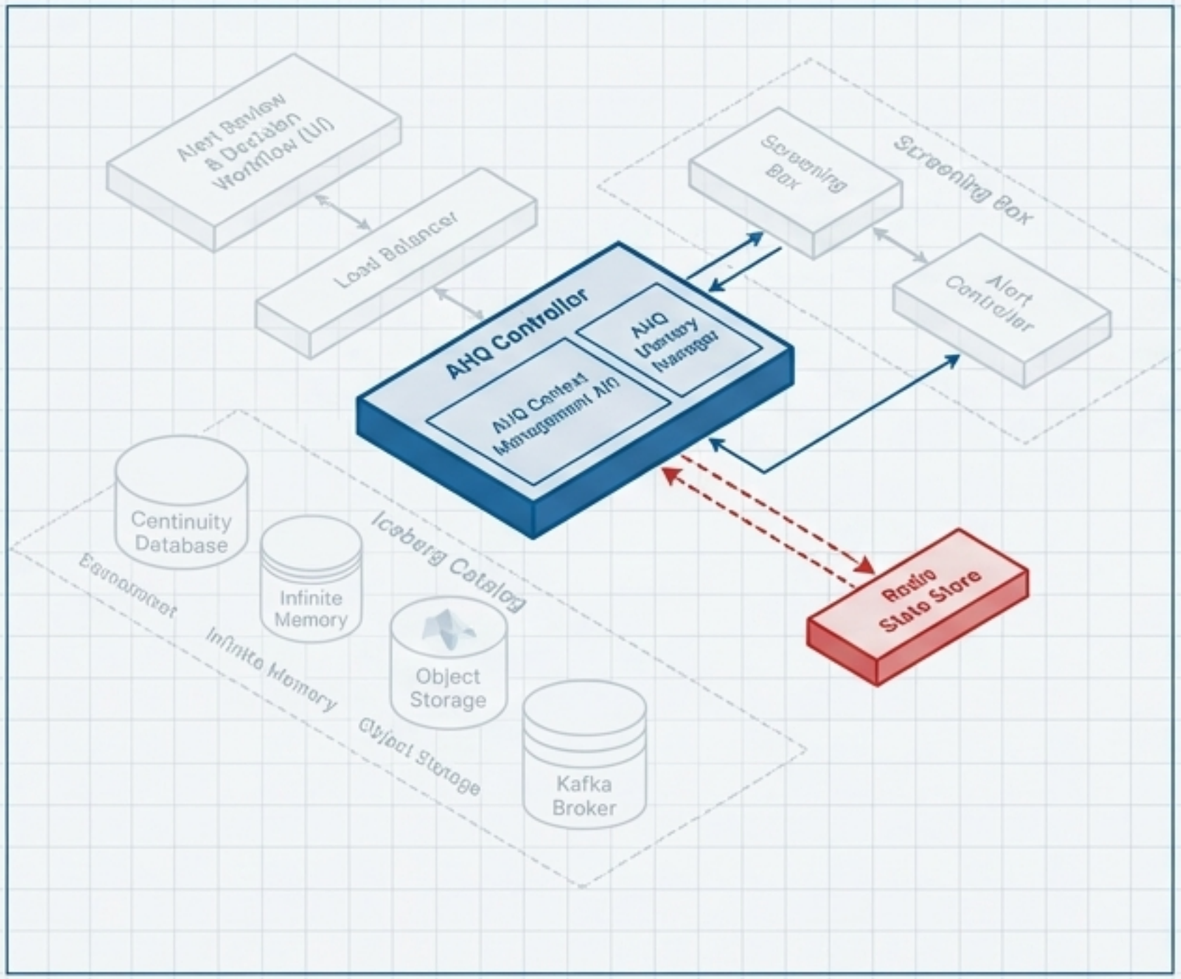
Component Breakdown

- **Core Product**: Orchestrator, DB Client (for database interaction).
- **Alert Review and Decision Workflow API Injector**: Mandatory plugin for high-throughput injection, required by ERF, AHQ, and Prediction Integrator.
- **Decision Reapplication**: Automatically applies previously made decisions to new, identical hits.
- **Pairing Manager**: Groups related alerts together for streamlined review.
- **Workflow Accelerator**: Applies rules to automate steps in the review process.
- **Infinite Memory Connectors**: Plugins for extracting and sending data to Infinite Memory.
- **Case Manager API Plugin**: Integrates with external case management systems.

Key Configuration Files: `AlertController/.env`, `AlertController/.env_AlertReviewApi`.

Deep Dive: The AHQ Controller - Centralized Intelligence

The Automated Hit Qualifier (AHQ) Controller is a dedicated service that manages the state and context required for machine learning-based hit disposition. It works in tandem with the AHQ plugins in the Screening Box and Alert Controller.



Component Breakdown

Core Product: Orchestrator for managing AHQ-specific flows.

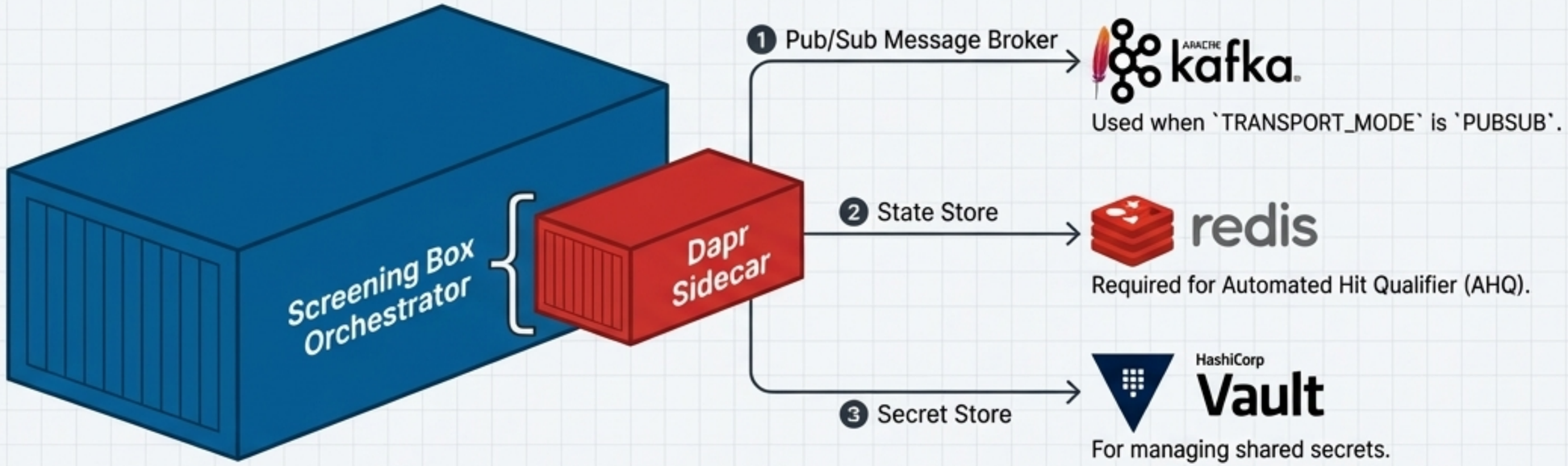
AHQ Memory Manager: The core plugin that interacts with the Dapr state store (e.g., Redis) to maintain real-time context for decisioning.

AHQ Context Management API: An optional but recommended plugin that allows users to manage AHQ contexts directly from the Alert Review front-end.

Architectural Note: The AHQ Controller's reliance on a state store (e.g., Redis, Oracle, SQL Server via Dapr) is a critical infrastructure consideration for deployment planning. This component is essential for both Production and Learning modes.

The Communication Fabric: Dapr Integration

Dapr is used as a sidecar to provide a consistent, reliable mechanism for inter-component communication and state management. It is not a standalone product but an integrated part of the Firco Continuity architecture.



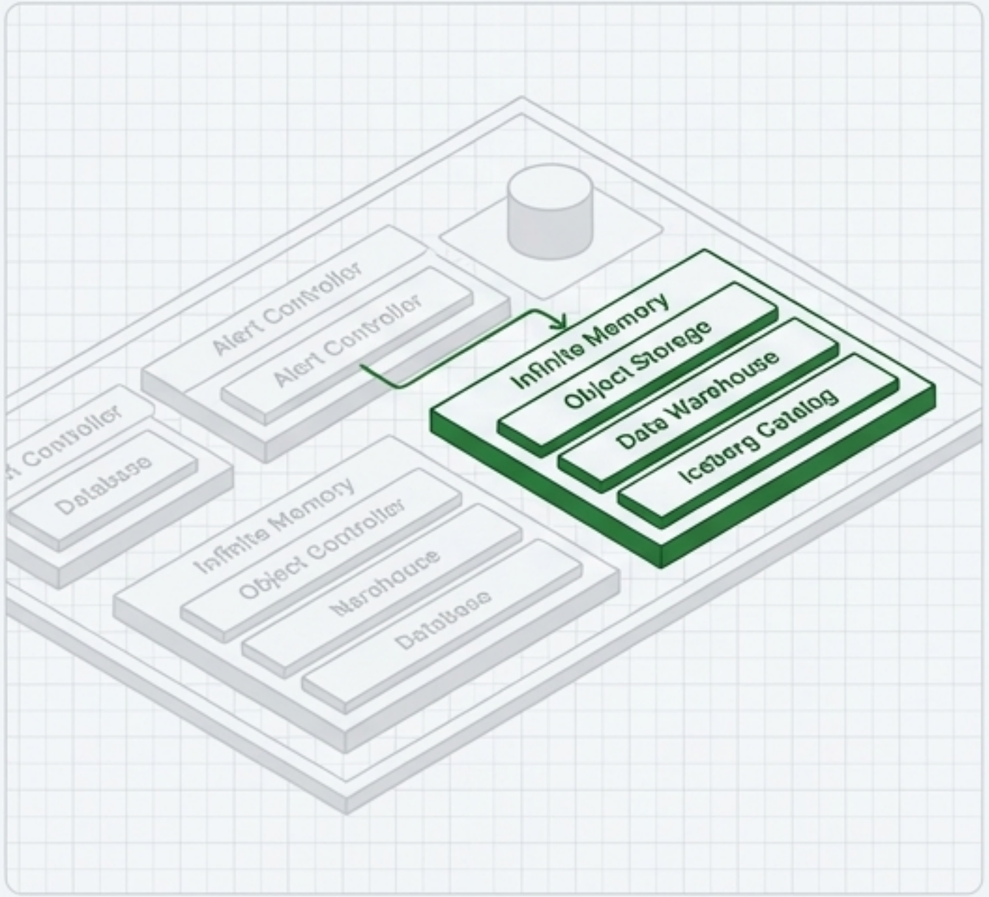
Key Dapr Features Used

- **Publish & Subscribe:** Decouples the Screening Box and Alert Controller for asynchronous, scalable communication.
- **State Management:** Provides a key/value store API for AHQ to persist hit context and model data.
- **Resiliency:** Configuration files (`resiliency.yaml`) define retry policies for communication, enhancing system stability.

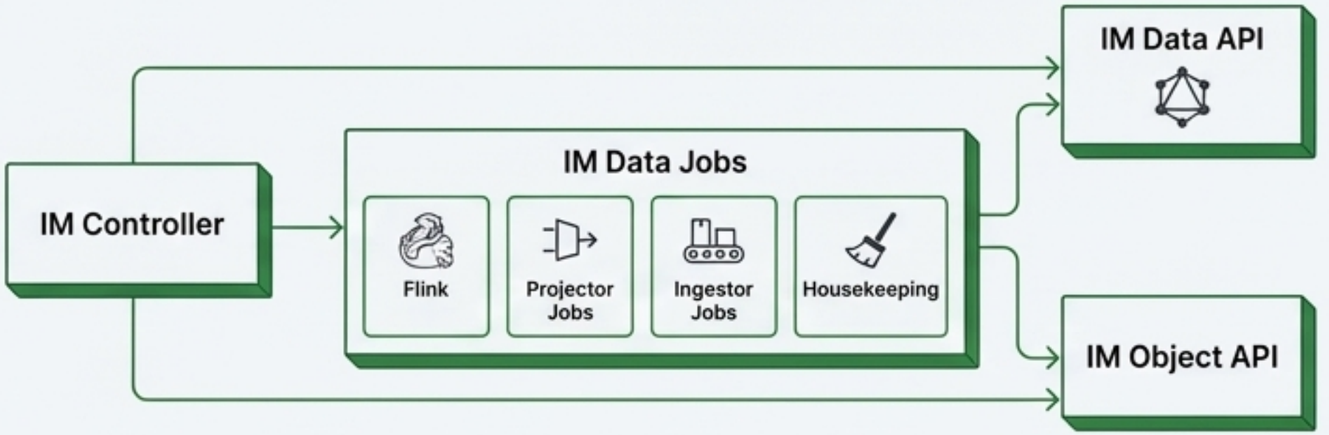
Installation Note: Dapr must be installed separately using the provided package. The start scripts for the core components are pre-configured to launch the Dapr sidecar when necessary.

Layer 3: The Archive - Infinite Memory Architecture

Infinite Memory provides a scalable, long-term archiving solution, offloading historical data from the primary operational database. It uses an Apache Iceberg-based data lake architecture for efficient storage and query performance.



Component Architecture

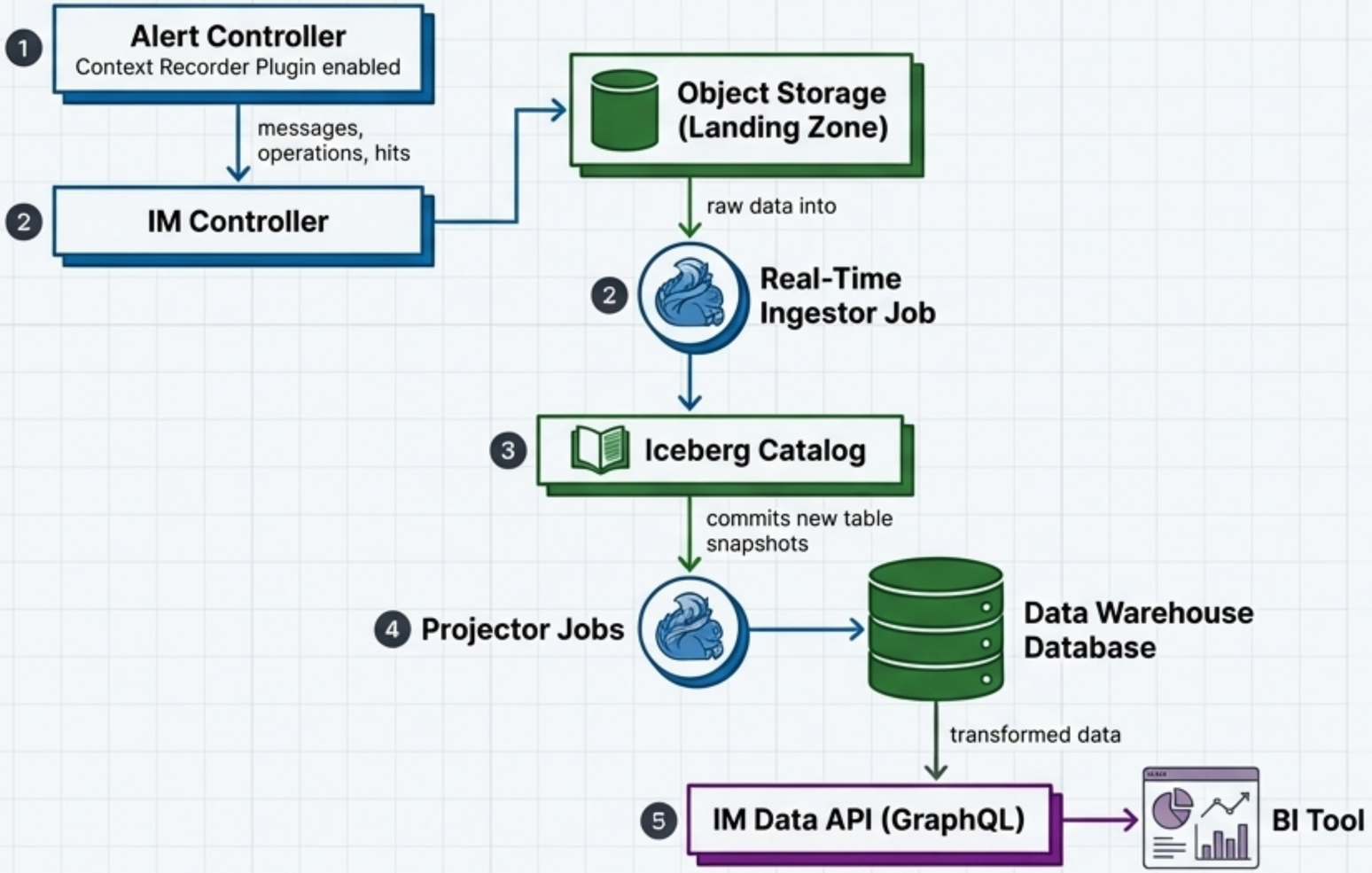


Underlying Technology Stack

- **Catalog**: Apache Iceberg JDBC Catalog (stores table metadata).
- **Storage**: Object Storage (S3-compatible) or Network File System (NFS).
- **Data Warehouse**: Relational database (Oracle/SQL) for projected data, optimized for reporting.

Infinite Memory: Data Ingestion and Processing Flow

Data is captured in **real-time** and ingested into the Infinite Memory data lake through a series of orchestrated jobs. Archives from older Firco versions (V5/V6) can also be loaded in batches.

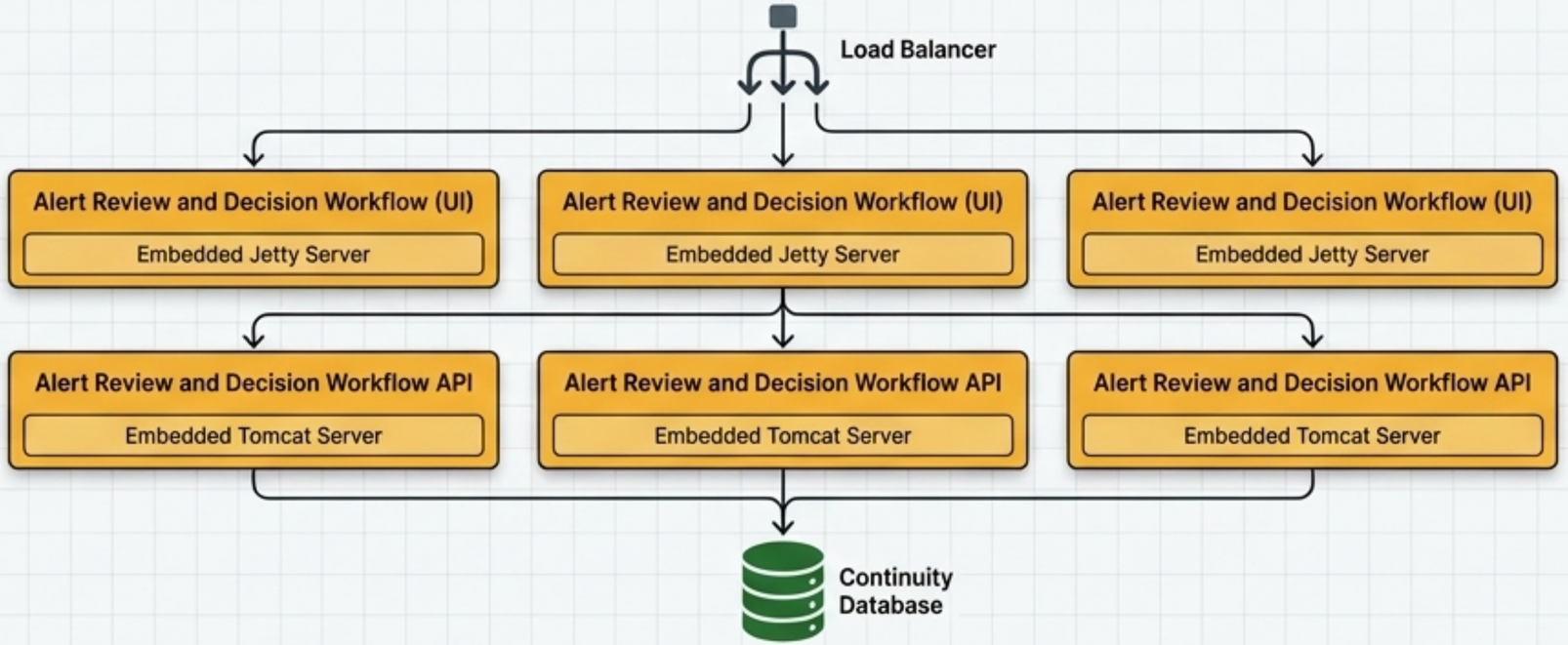


Key Jobs and Scripts

- `start_realtime_ingestor`: For continuous, implicit archiving.
- `start_archiveV5_ingestor` / `start_archiveV6_ingestor`: For batch loading historical archives.
- `start_projector_realtime`: To populate the data warehouse.
- `start_im_data_maintenance_...`: Scripts for housekeeping tasks like expiring old snapshots and deleting orphan files.

Layer 4: The Interface - Front-End Architecture

The front-end is delivered as two distinct Java applications: the user interface itself and the backing API. This separation allows for independent scaling and security. Apache Tomcat is embedded within the packages, simplifying deployment.



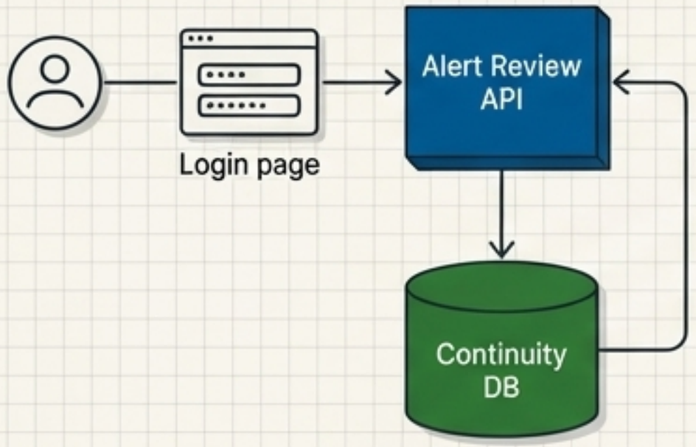
Key Architectural Concepts

- **Stateless Application:** The front-end is stateless, simplifying load balancing as no session persistence or sticky sessions are required.
- **Health Checks:** Each component exposes health check endpoints (`/actuator/health`) for load balancer monitoring.
- **Independent Scaling:** The UI and API can be scaled independently based on load. For example, more API instances might be needed to handle high volumes of automated tasks.
- **Configuration:**
  - UI: `ContinuityAlertReview.yml`
  - API: `ContinuityAlertReviewApi.properties`

Layer 5: The Gatekeeper - Security and Authentication

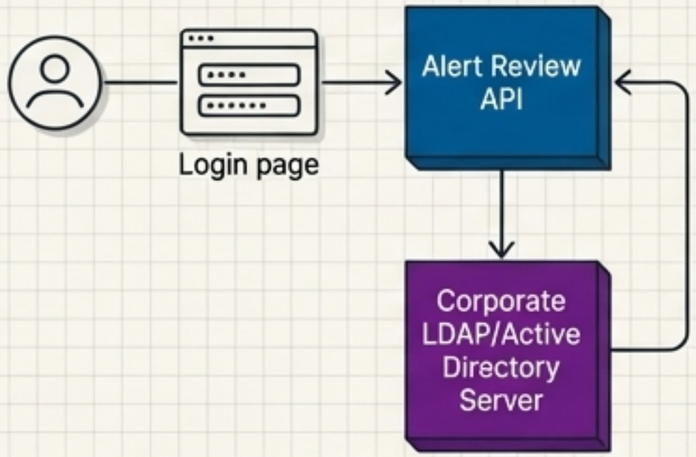
The platform supports multiple authentication models to integrate with enterprise security standards. Authentication is configured in the `ContinuityAlertReviewApi.properties` file.

Default Authentication



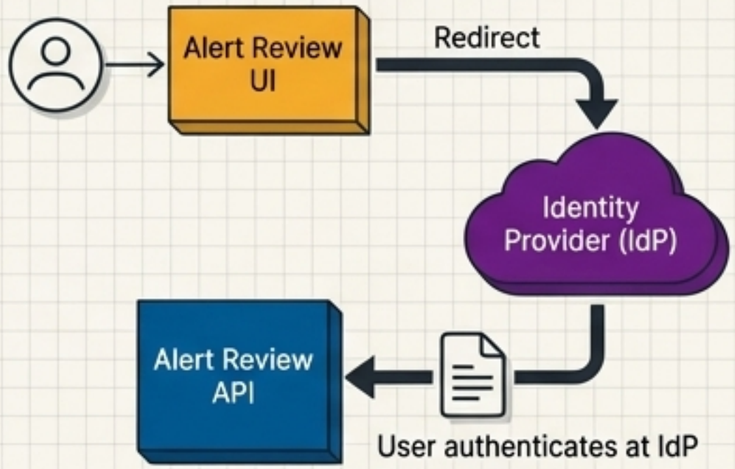
Built-in user management. Credentials stored and managed within the Firco Continuity database.

LDAP Authentication



Delegates authentication to an existing LDAP or Active Directory server. Users are still managed in Continuity.

SAML 2.0 (SSO)



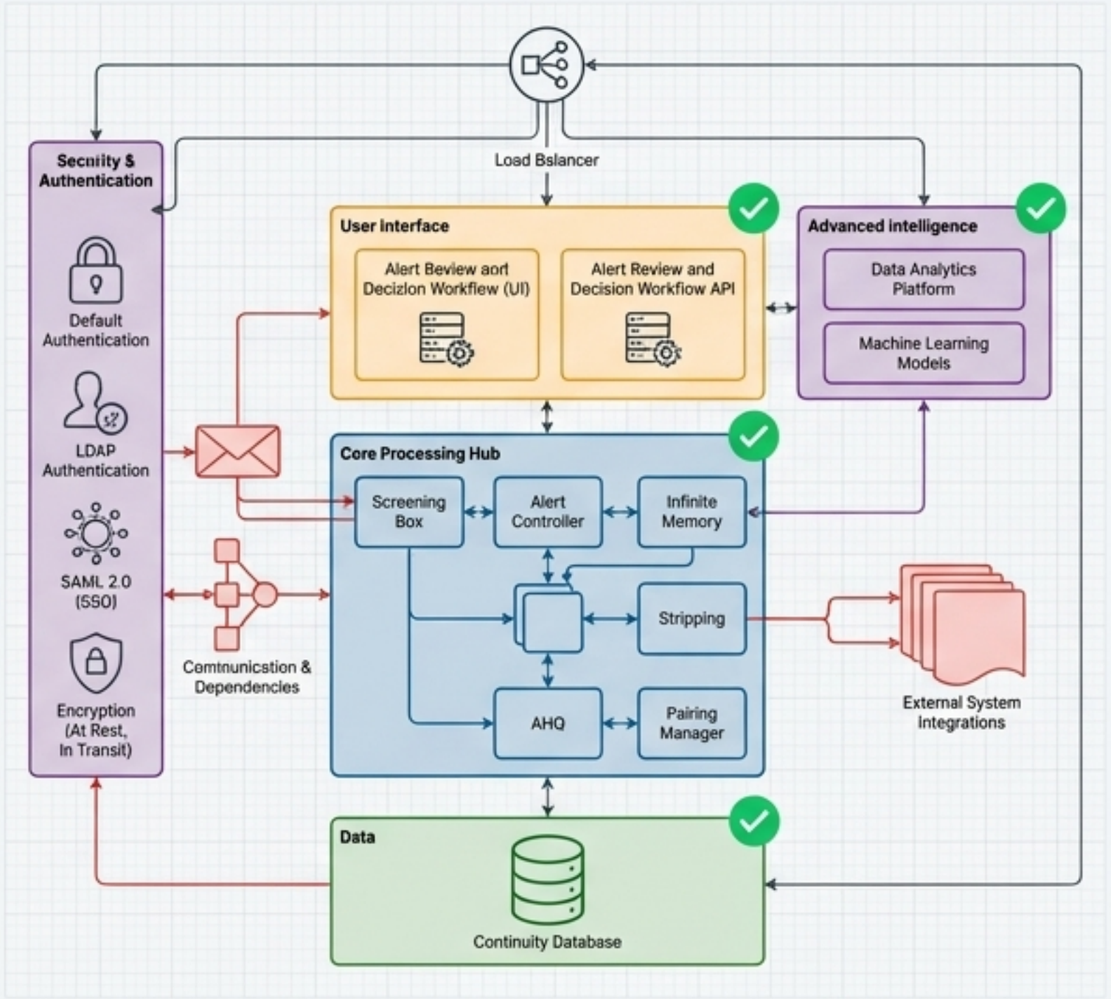
Enables Single Sign-On (SSO) by integrating with a SAML 2.0 Identity Provider like ADFS or Entra ID.

Advanced Feature: SAML User Provisioning

When enabled (`saml.user-provisioning.enable=true`), the IdP becomes the source of truth for user identity, profiles, and unit assignments. This centralizes user management and automates provisioning and de-provisioning.

The Complete Blueprint: A Modular, Scalable, and Secure Ecosystem

By assembling these layers—from the foundational database to the intelligent processing core and the user-facing interface—the Firco Continuity architecture provides a comprehensive solution for real-time compliance screening.



Summary of Architectural Principles

Modularity: Independent components (Screening Box, Alert Controller, Infinite Memory) allow for flexible deployment, scaling, and maintenance.

Configurability: Behavior is controlled through externalized environment files (`.env`, `.properties`, `.yml`), enabling adaptation to different environments without code changes.

Scalability: The architecture supports horizontal scaling of all major components, from multiple Screening Box instances to a load-balanced front-end, to handle growing transaction volumes.

Extensibility: Plugin-based design for components like Stripping, AHQ, and Pairing Manager allows for the addition of new capabilities.

Security: Integrated support for enterprise authentication standards (LDAP, SAML) and robust encryption options for securing data at rest and in transit.