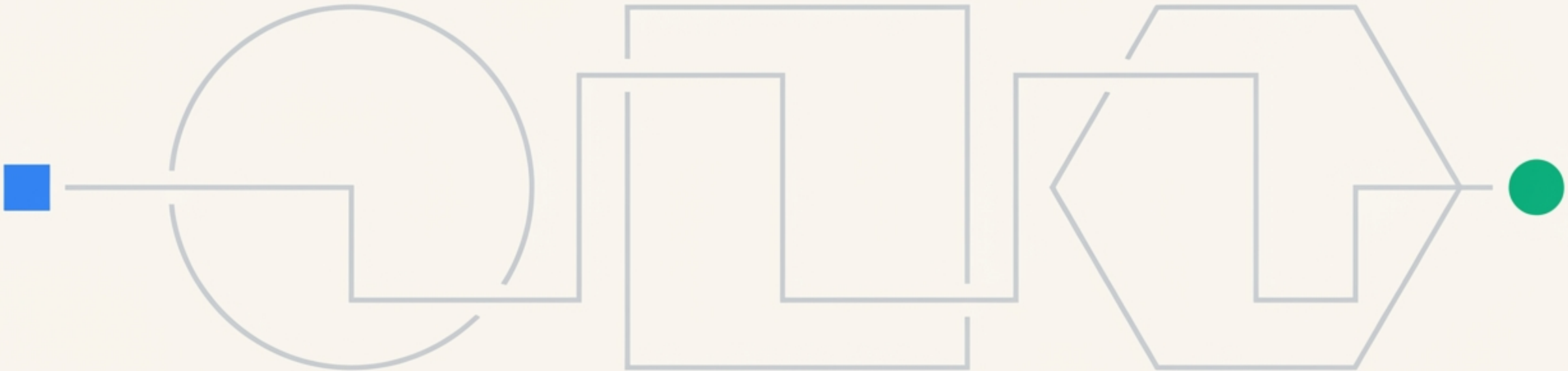
The Journey of a Transaction

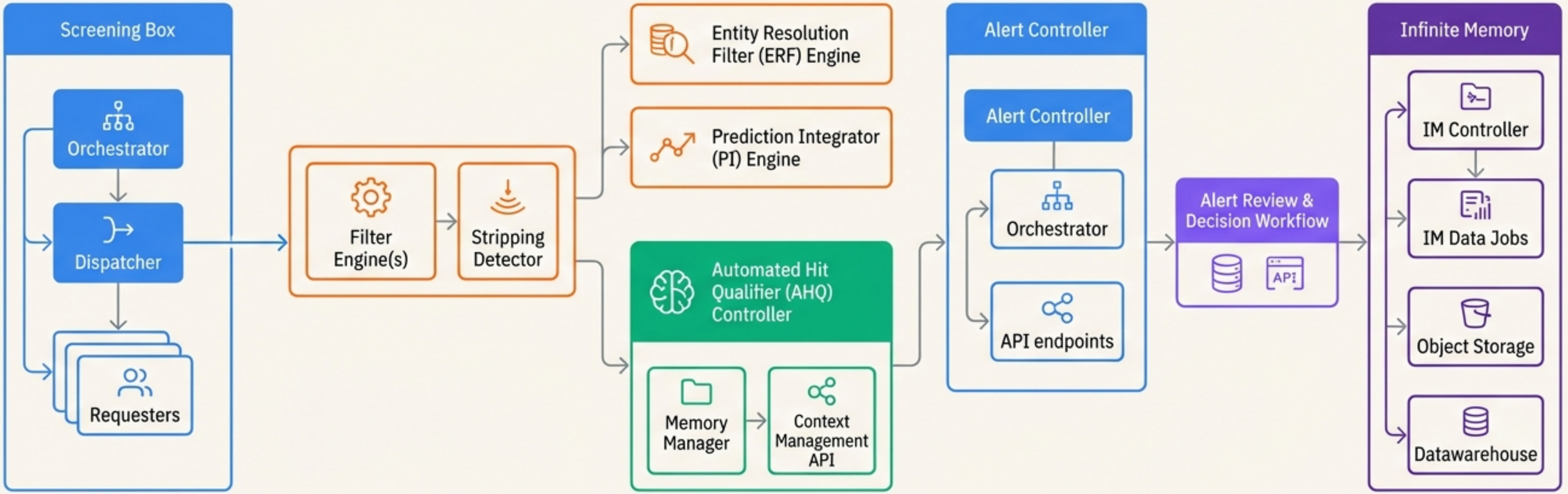
An End-to-End Architectural Flow in LexisNexis® Firco™ Continuity



NotebookLM

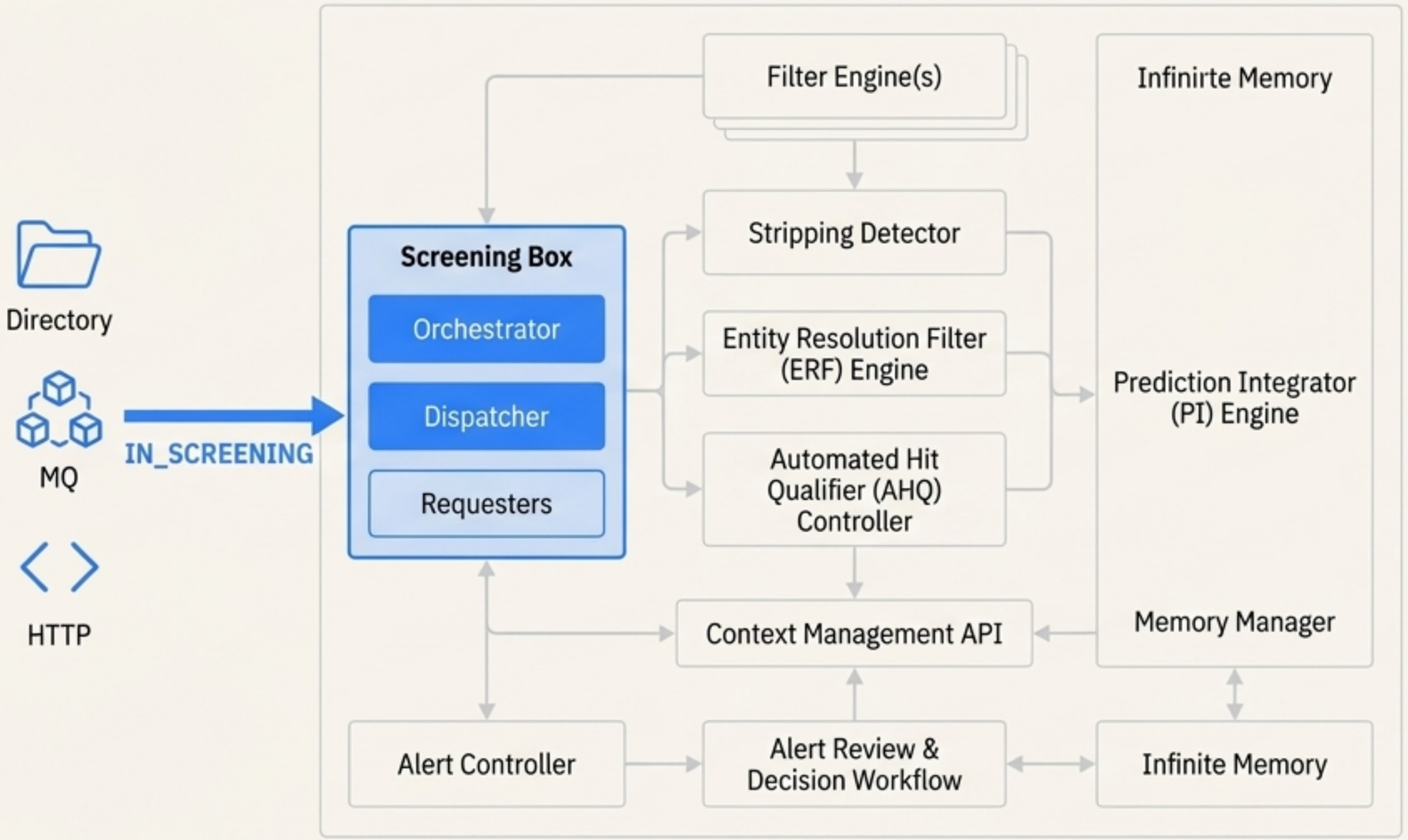
The Firco Continuity Ecosystem: Our Journey's Map

This presentation follows a single transaction message from the moment it enters the system, through screening and decisioning, to its final disposition. This diagram serves as our map. In the following slides, we will zoom in on each stage of the journey.



NotebookLM

Stage 1: Ingress and Normalization



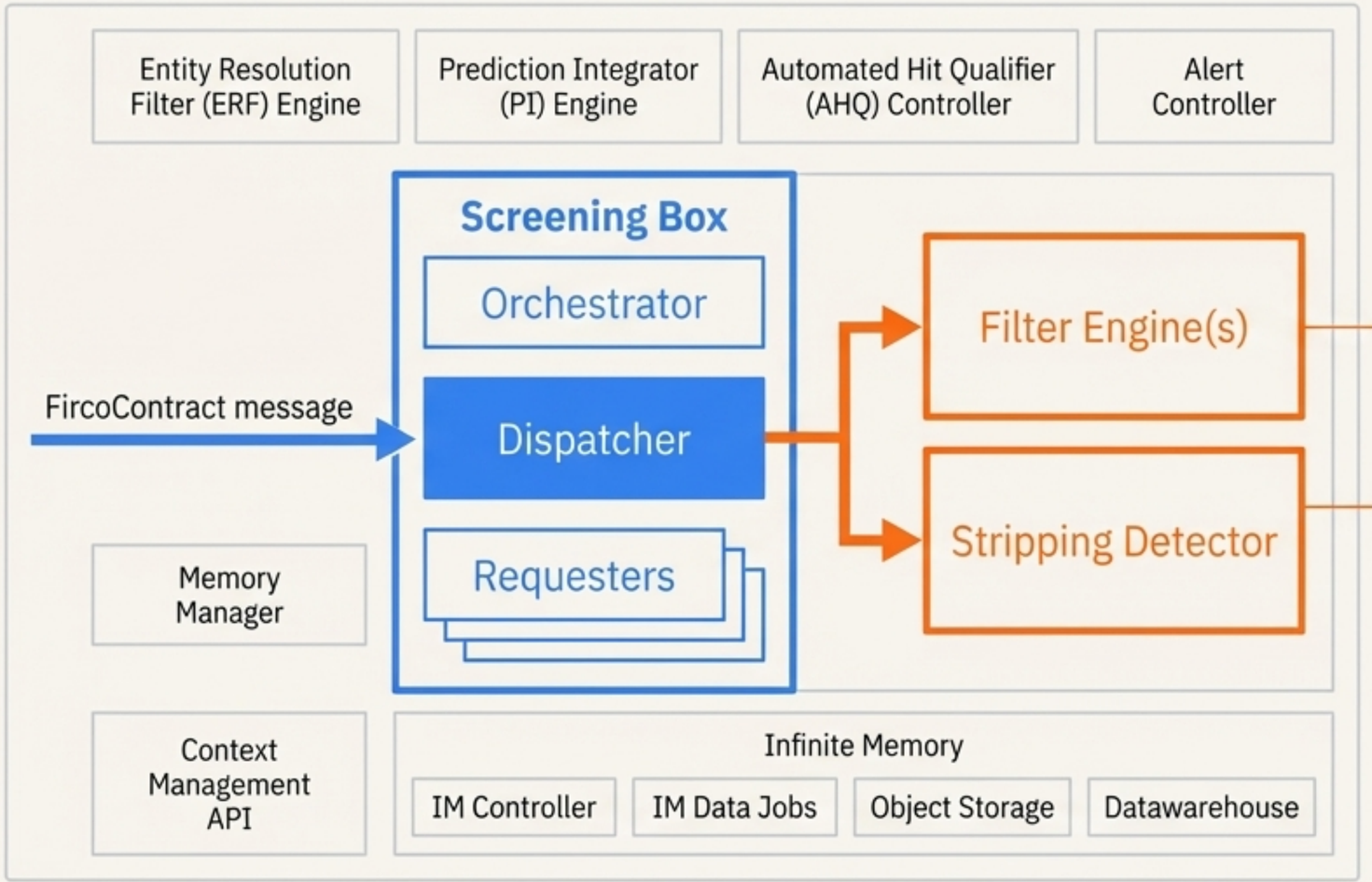
1. **Ingestion:** An incoming message arrives at the Screening Box via a configured transport mode (e.g., Directory, MQ, HTTP).
2. **Parsing & Mapping:** The Orchestrator engine uses a `parser` (e.g., JSON) and a `mapper` (e.g., `"FircoContract(innerMapper: swiftMasterLegacy, innerParser: SwiftFIN)"`) to transform the raw message into the standardized `FircoContract` internal format.
3. **Metadata Initialization:** If the incoming message lacks key metadata, the system populates it with default values to ensure consistent processing.

Missing Metadata	Default Value
BusinessUnit	ROOT
MessageID	Automatically generated
Unit	WMQ
ApplicationCode	APP_WMQ
Amount	0.00
IOIndicator	I

NotebookLM

Stage 2: The Screening Gauntlet

Once normalized, the message is passed to the Screening Box Dispatcher. This FCL (Firco Common Language) script orchestrates the analysis by executing a series of sequential and conditional screening steps.



Step 1: Core Filtering

The dispatcher first calls `screen msg using FILTERING_REQUESTER`. The message body is sent to the Filter Engine, which screens it against sanction lists and compliance rules. Any resulting hits are appended to the message.

Step 2: Stripping Detection

The dispatcher then calls `screen msg using STRIPPING_REQUESTER`. The message is analyzed for potential modifications or 'stripping' of sensitive information. Stripping alerts are added if detected.

NotebookLM

Under the Hood: The Screening Box Dispatcher Logic

screening-box-dispatcher.fcl

```
// Default metadata initialization
if length(msg/Metadata/Unit) = 0
   "WMQ" -> msg/Metadata/Unit
// ... other metadata rules

// Core Screening (Sequential)
1  screen msg using FILTERING_REQUESTER(...)          // <-- Annotation 1
2  screen msg using STRIPPING_REQUESTER               // <-- Annotation 2

// Advanced Analysis (Conditional)
3  if ERF and (call IsGloballyEligible on msg) ...    // <-- Annotation 3
      screen msg using ERF_REQUESTER

4  if AHQ and call IsGloballyEligible on msg          // <-- Annotation 4
      screen msg using AHQ_REQUESTER

5  if PI and (call IsGloballyEligible on msg) ...     // <-- Annotation 5
      screen msg using PREDICTION_INTEGRATOR_REQUESTER

// Finalization & Routing
6  apply StatusMsgResolution on msg                   // <-- Annotation 6
7  route msg to NOTIFICATION                          // <-- Annotation 7
8  route msg to OUT_INTERNAL_ALERT_CONTROLLER         // <-- Annotation 8
```

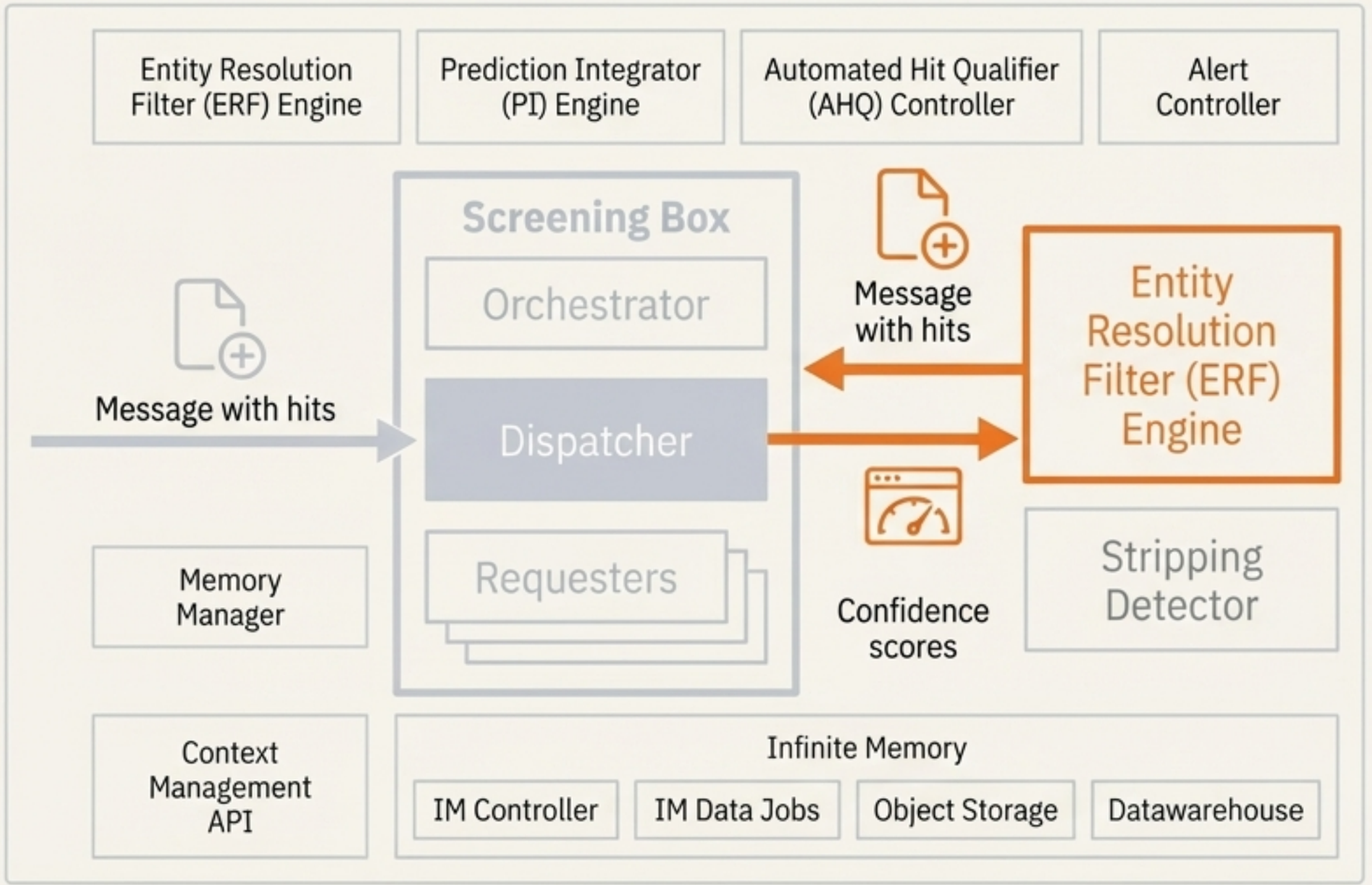
1. Primary list screening
2. Stripping detection
3. Check eligibility for ERF and call if true
4. Check eligibility for AHQ and call if true
5. Check eligibility for PI and call if true
6. Calculate final status
7. Route to external notification
8. Route to Alert Controller

Key Insight: The dispatcher's power lies in its combination of mandatory sequential calls (Filter, Stripping) and flexible, conditional calls to advanced engines (ERF, AHQ, PI), all governed by FCL rules.

NotebookLM

Stage 3a: Enrichment via Entity Resolution Filter (ERF)

If the message is eligible (determined by the `IsEntityResolutionFilterEligible` function), the dispatcher sends it to the Entity Resolution Filter (ERF) to reduce blocking hits by analyzing entity data in context.



How it Works

1. **Request:** The `ERF_REQUESTER` extracts relevant data (names, locations, BIC/IBANs from MT or ISO 20022 messages) and sends it to the ERF tool.

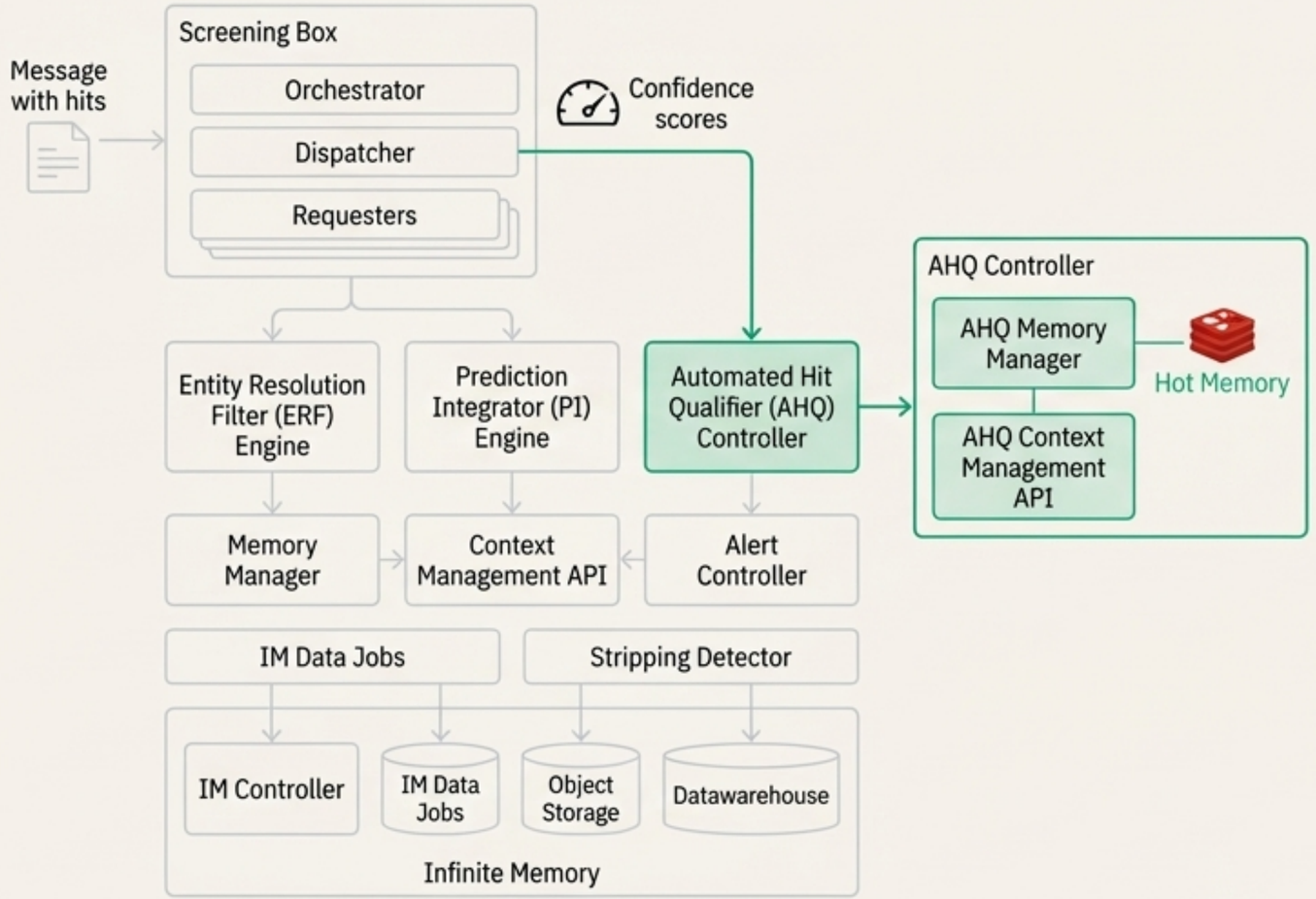
2. **Scoring:** ERF analyzes the context and returns a confidence score for each hit.

3. **Post-Processing:** The results are processed by `ERFResponseRules.fcl`. Based on the scores, FCL rules can automatically change a hit's status from `BLOCKING` to `NON_BLOCKING`, adding a corresponding reason code.

NotebookLM

Stage 3b: Automation with Automated Hit Qualifier (AHQ)

For eligible messages, the AHQ_REQUESTER consults the Automated Hit Qualifier to see if a recurring hit can be automatically resolved based on past analyst decisions.



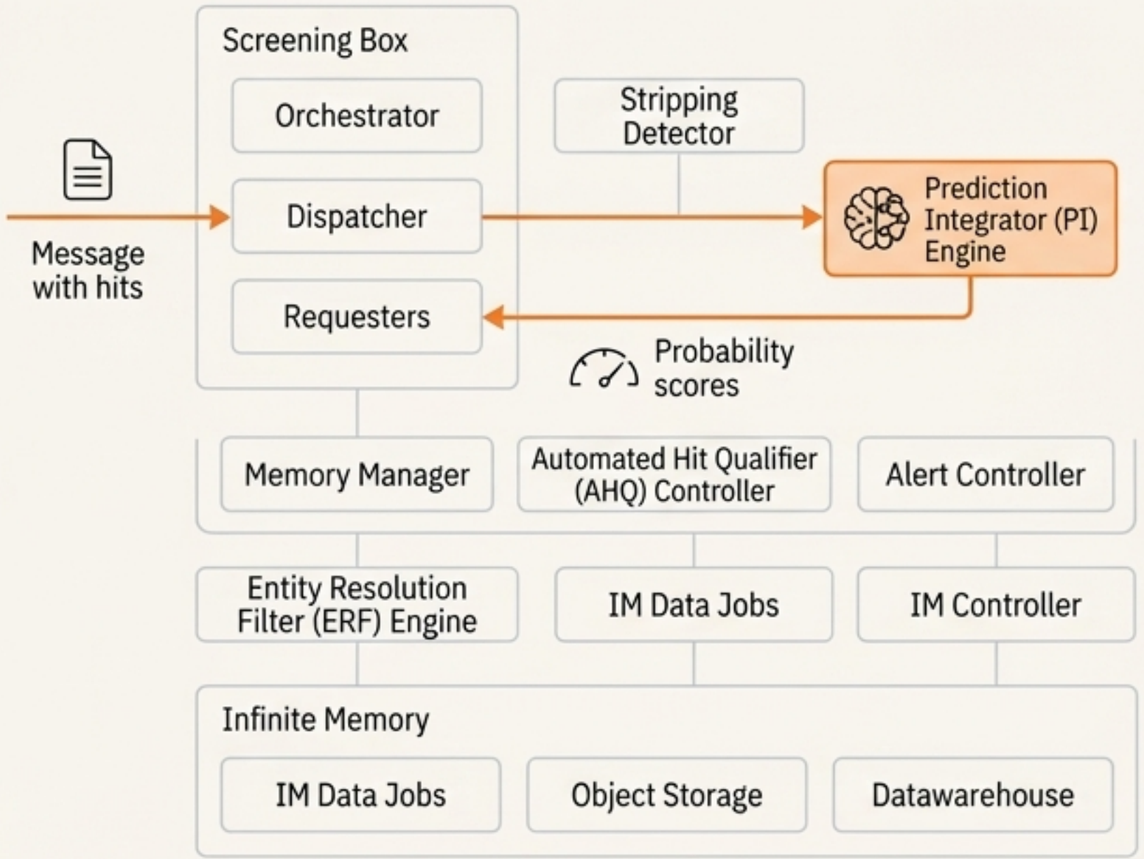
How it Works

1. **Context Creation:** When an analyst first qualifies a new type of hit, an AHQ rule (`ahq.rule.yaml`) creates a 'context' based on a signature (e.g., entity ID, matching text, message fields). This context is stored.
2. **Real-Time Check:** For new messages, the requester sends the hit's signature to the AHQ Memory Manager.
3. **Automatic Qualification:** If a matching context is found and is in an `` `ENABLED` `` state (meaning analysts have consistently resolved it as a false positive in the past), AHQ automatically qualifies the new hit, reducing manual review.

NotebookLM

Stage 3c: Predictive Scoring via Prediction Integrator (PI)

Prediction Integrator connects Firco Continuity to an external prediction engine (AI/ML model) to score the probability that a hit is a true positive, further reducing false positives.



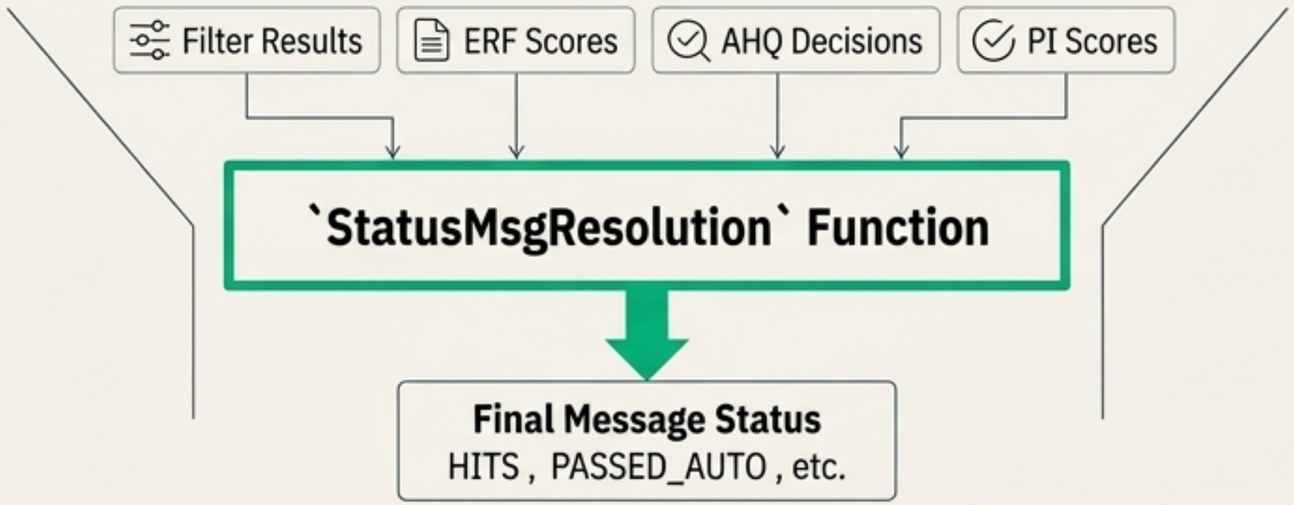
How it Works

1. **Eligibility Check**: The **IsPredictionIntegratorEligible** function determines if the message should be sent for scoring. Stripping hits are never sent.
2. **API Call**: The PREDICTION_INTEGRATOR_REQUESTER sends the alerted message to the configured API endpoint.
3. **Receive Score**: The external engine returns a probability score for the message and/or individual hits.
4. **Apply Results**: FCL rules within **PredictionIntegratorRules.fcl** process the score. For example, a rule can change a message's status to NON_BLOCKING if its score is above a certain threshold (e.g., > 80).

NotebookLM

Stage 4: Calculating the Verdict

The **`StatusMsgResolution`** function is the final step within the dispatcher. It consolidates results from all preceding engines to calculate the definitive status of the message before it leaves the Screening Box.



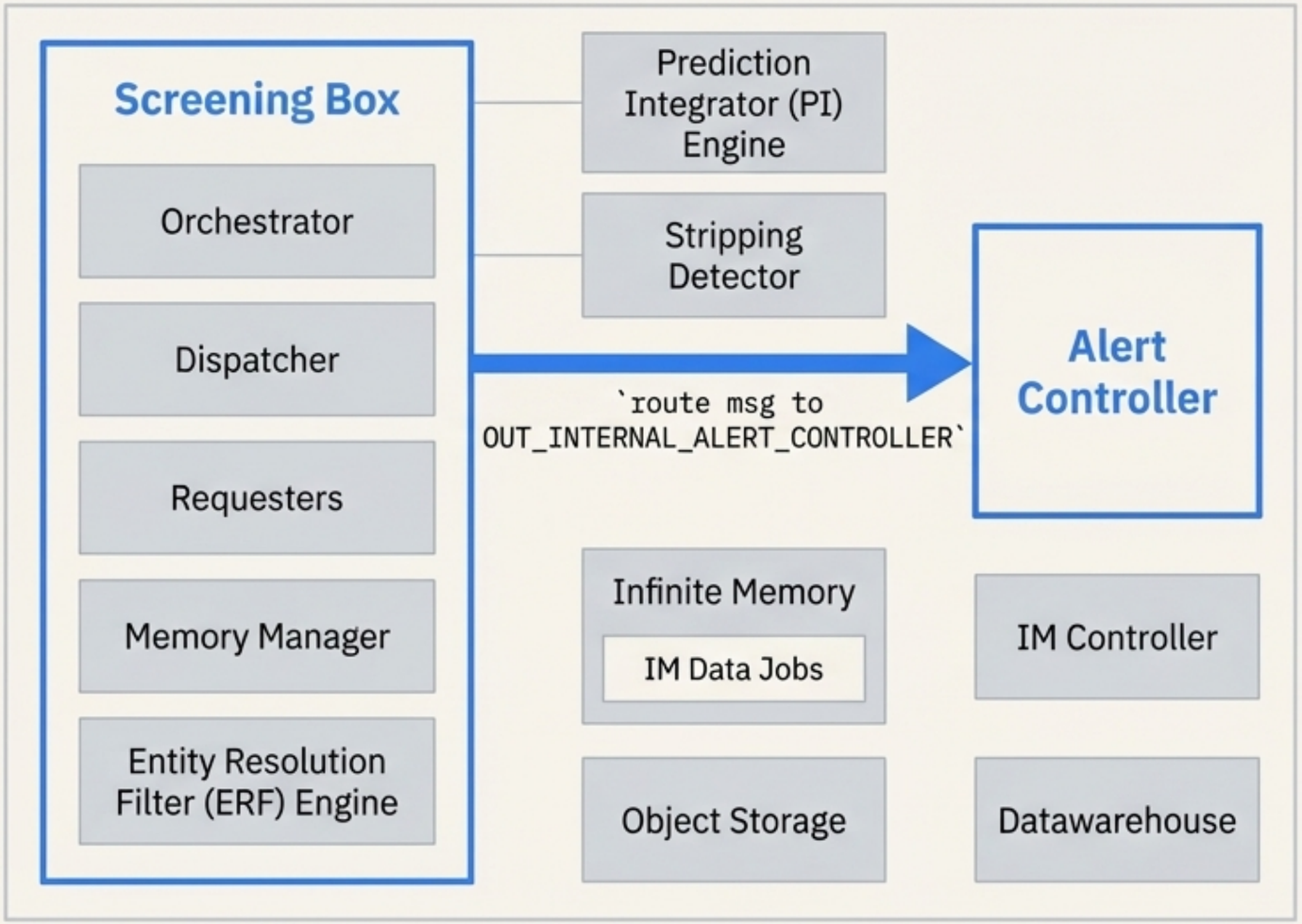
Calculation Order

The status is calculated based on a strict hierarchy.

If Message Contains...	Resulting Status
At least 1 blocking hit	**HITS**
0 blocking hits, AND at least 1 hit was qualified as false (by AHQ, ERF, PI)	**PASSED_AUTO**
0 blocking hits, AND at least 1 non-blocking hit (status 2)	**NON_BLOCKING**
All hits were ignored (status 3 or 4)	**HIT_IGNORED**

NotebookLM

Stage 5: Handover for Remediation

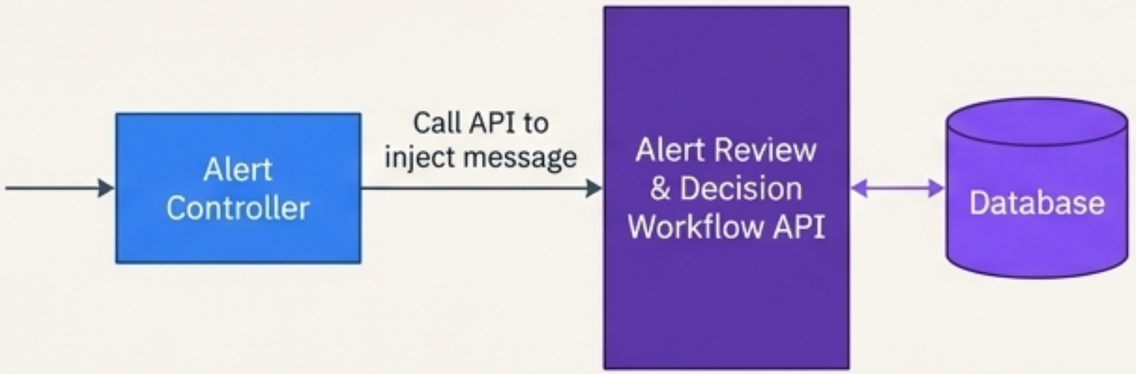


With its screening complete and status calculated, the message is routed to the Alert Controller. This component is responsible for inserting alerted messages into the database and managing notifications.

- **Mandatory Format:** The communication between Screening Box and Alert Controller *must* use the `FircoContractVx` format. This ensures that all screening results, hit details, statuses, and metadata are transferred reliably.
- **Endpoint Configuration:** The Screening Box `OUT_INTERNAL_ALERT_CONTROLLER` producer is configured to send to the Alert Controller's `INTERNAL_ALERT_CONTROLLER` consumer endpoint.

NotebookLM

Stage 6: Persistence and Alert Generation



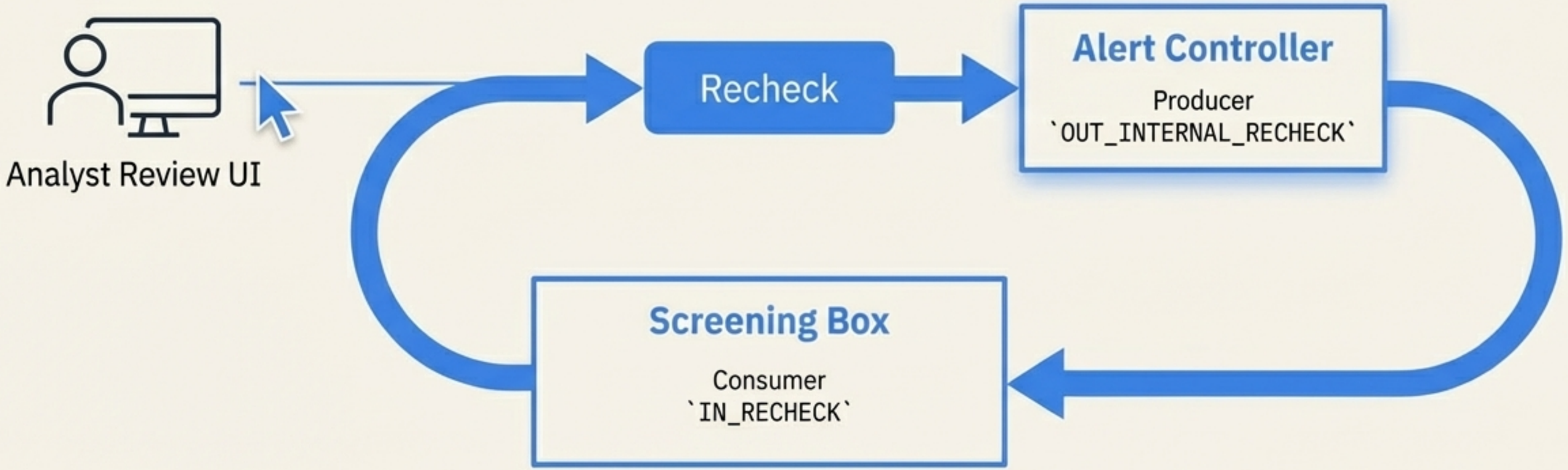
Upon receiving the message, the Alert Controller's primary role is to persist it. It calls the Alert Review and Decision Workflow API to inject the message and its associated hits into the database, making it available for analyst review.

HTTP Code	Meaning	Orchestrator Behavior
200	Success	Acknowledges and completes.
400	Bad Request (e.g., missing mandatory data)	Routes message to an error queue.
500	Server Error (e.g., context doesn't exist)	Retries the request.

NotebookLM

Stage 7: Analyst Review and the Recheck Loop

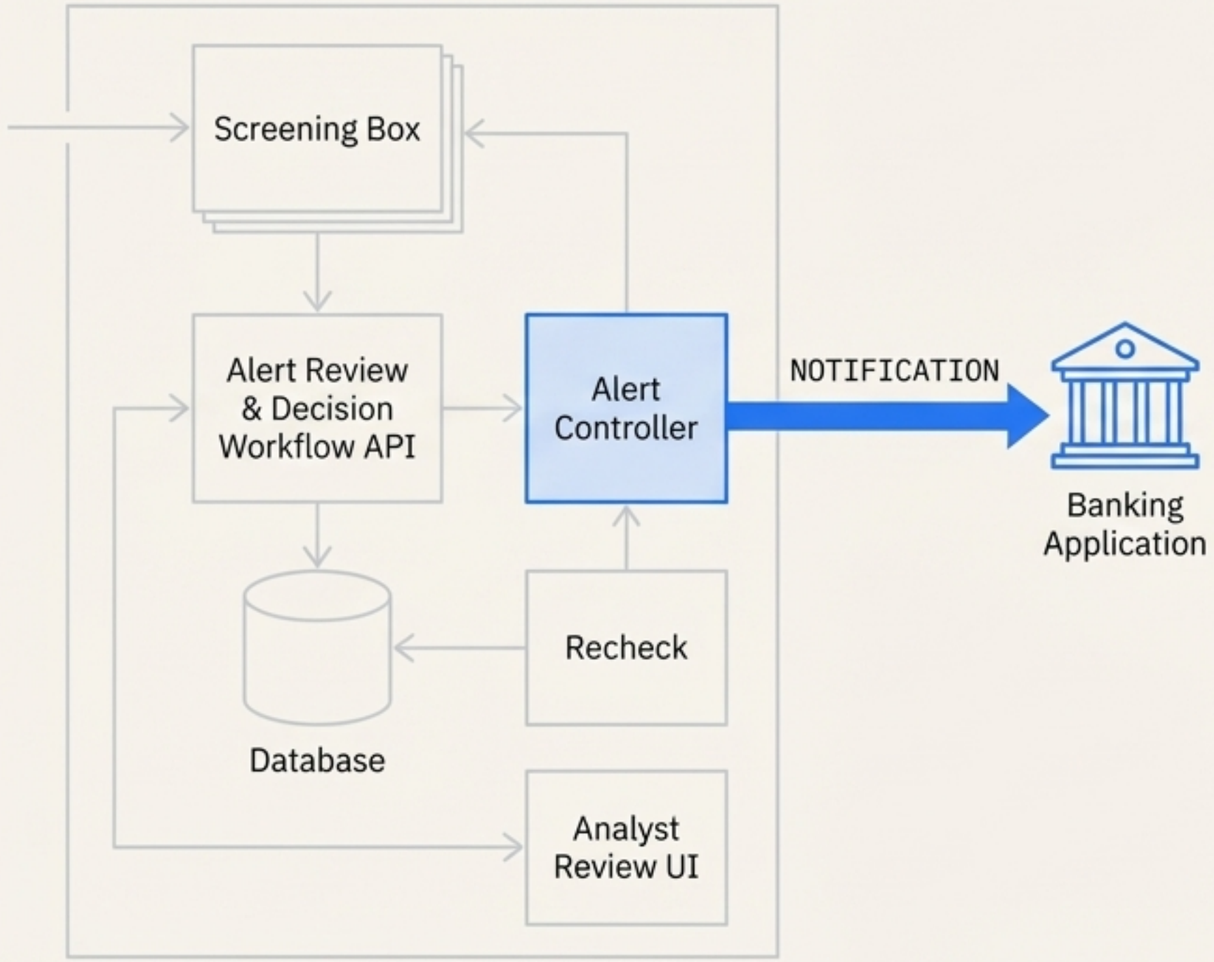
Analysts review alerted messages in the Alert Review and Decision Workflow UI. If a change is made (e.g., to a filter list) that might affect the outcome, an analyst can trigger a “Recheck”.



The recheck feature sends the message from the Alert Controller back to the Screening Box to be processed again with the latest configuration. This re-enters the message at Stage 1 of its journey. (Note: Rechecked messages are processed as new messages by AHQ).

NotebookLM

Stage 8: Egress and Final Notification



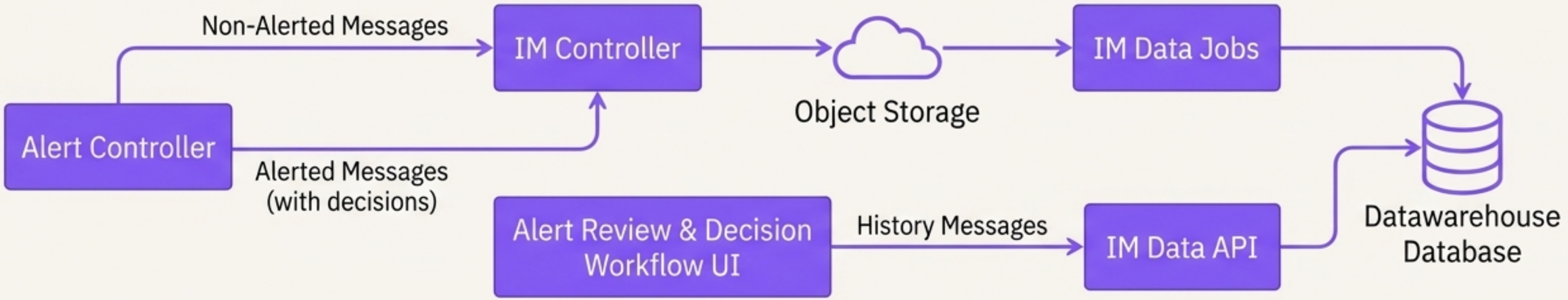
Once a final decision is made on a message (either automatically or by an analyst), the Alert Controller's `EVENT_PUSH_FLOW` sends a final status notification back to the originating banking application, completing the transaction screening lifecycle.

- The `NOTIFICATION` producer in the Alert Controller handles this egress.
- The output format can be configured using mappers:
  - "`WSJWMQOUT_V2`": The default format, compatible with the legacy DB Client extractor.
  - "FircoContractVx.Message": A richer format that includes full traceability with all events, hits, and entities.

NotebookLM

Stage 9: Archiving in Infinite Memory

Infinite Memory provides a scalable solution for long-term data storage. It decouples active processing from historical data, improving performance and enabling massive data retention.

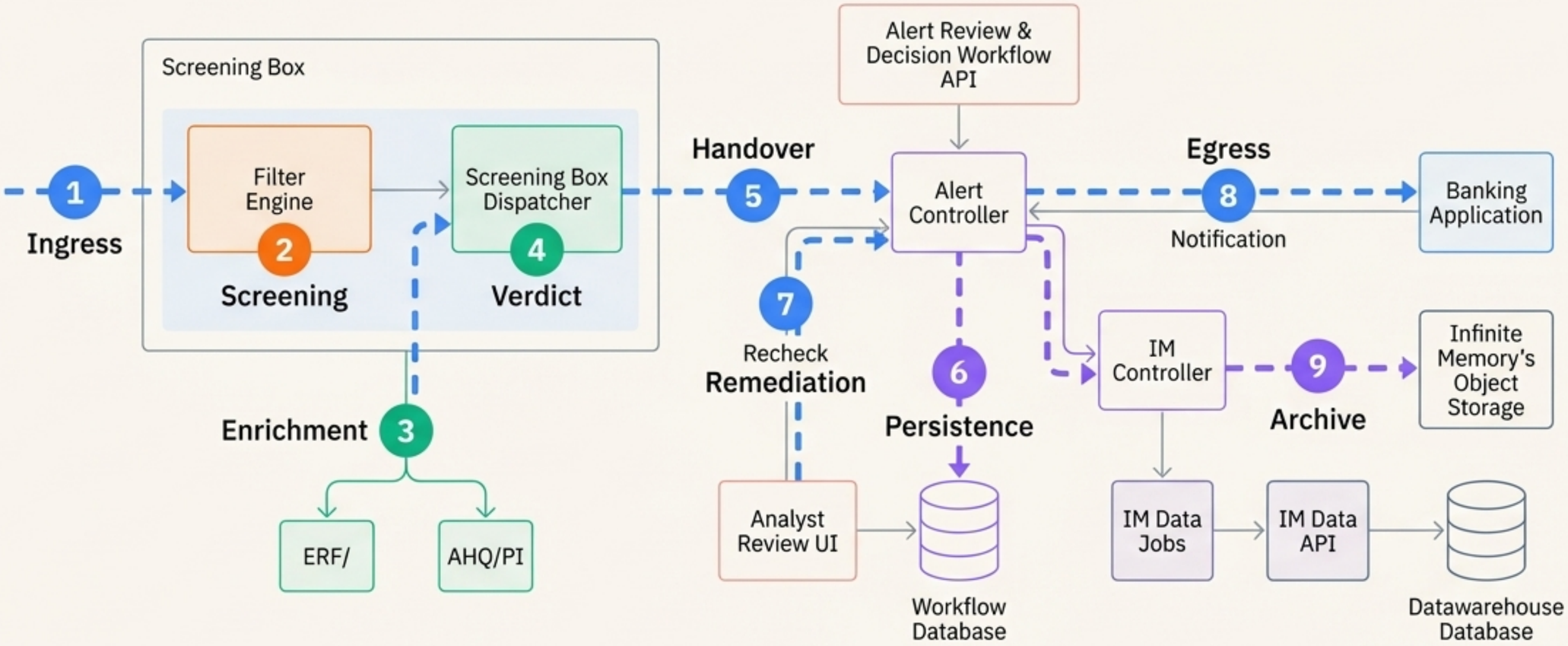


Key Concepts

- **Non-Alerted Messages**: Go directly to object storage, bypassing the main transactional database.
- **Alerted Messages**: Are stored after a final decision is made.
- **Ingestion & Projection**: `IM Data Jobs` transforms the raw data from object storage (ingestion) and projects a queryable subset into a datawarehouse for reporting and historical views (projection).

NotebookLM

The Message’s Journey, Complete



From raw transaction to final notification and long-term storage, the Firco Continuity back-end provides a robust, orchestrated, and extensible platform for transaction screening. Each component plays a precise role in a dynamic, end-to-end process designed for accuracy, efficiency, and scale.

NotebookLM