Economic Calendar with AI Impact (EC)

Technical Design Document v1.0 (MVP & Phase 2 – Engineering Edition)

Prepared by: Goldh Technical Team
Date: November 2025
Related Modules: UMF, GID, TAC
Document Type: Developer-Facing | Architecture Reference
Review Status: For Internal Engineering Review and Approval

---

1. Overview and Scope

1.1 Purpose

The **Economic Calendar (EC)** module provides macroeconomic event intelligence and AI-assisted impact assessment across asset classes.
It acts as a contextual layer for the Goldh platform, correlating market reactions to global events like CPI releases, FOMC minutes, or crypto regulation announcements.

1.2 Scope

- Aggregate and normalize global macroeconomic calendars and earnings events.
- Apply an **AI Impact Engine** to estimate short-term and long-term asset impacts.
- Publish macro sentiment and event-triggered confidence adjustments for other modules (UMF, TAC, GID).

MVP (Free Tier): Pull-based ingestion from free macro APIs (EconoData, Investing.com, Alpha Vantage).
Phase 2 (Premium Tier): Real-time impact estimation using paid APIs (Trading Economics, Refinitiv, Benzinga Calendar) and ML-driven forecasting.

1.3 Intended Audience

Data engineers, backend developers, data scientists, and product teams working on analytics and event-driven integration across the Goldh stack.

1.4 References

- EC BRD v1.0
- Goldh API Master List Workbook
- UMF TDD v1.0 (6.5, 9.4, 12.6, 16.1)
- GID TDD v1.0 (9.4, 12.5, 14.5)
- TAC TDD v1.0 (4.2, 7.3)

---

2. System Architecture

2.1 Architecture Overview

The EC module is a multi-layered system combining event ingestion, impact modeling, and distribution pipelines.
It operates as a **core dependency** for contextualizing all market analytics across the Goldh platform.

Primary Components

1. **Event Collector:** Gathers and normalizes global macro events.
2. **AI Impact Engine:** Estimates magnitude and direction of asset impact.
3. **Storage Layer:** Persists historical events and model results.
4. **Distribution Layer:** Publishes enriched events to the Confidence Bus and other modules.

MVP Design

- Scheduled polling from free event APIs (JSON feeds).
- Rule-based impact mapping (event type → expected movement).
- Batch updates to Redis and S3.
- Limited backtesting capability.

Phase 2 Design

- Streaming ingest using WebSocket or webhook feeds from paid providers.
- Deep learning model predicts asset volatility response within 24h window.
- Integration with SageMaker for online inference.
- Interactive macro-impact heatmap for visualization in UMF dashboards.

Cross-Reference: See **UMF TDD 9.4** for asynchronous event pipeline integration with EC streams.

2.2 Component Breakdown

Component	MVP	Phase 2
Sources	EconoData, Alpha Vantage	Trading Economics, Refinitiv, Benzinga
Processing	Rule-based event parser	AI-driven volatility predictor
Storage	Redis + S3 snapshots	RDS (PostgreSQL) + Feature Store
Distribution	REST endpoints	Event Bus (Kinesis) + GraphQL API
Security	JWT Auth	mTLS + IAM roles

2.3 High-Level Flow

1. Event Collector polls macro and earnings APIs.
2. Parser standardizes event names, timestamps, and country codes.
3. Impact Engine calculates score using predefined rules (MVP) or ML inference (Phase 2).
4. Events are stored and broadcast to UMF, TAC, and GID.
5. UI displays calendar and predicted market sensitivity indicators.

Decision Paths:

- If event data incomplete → use prior period values.
- If event category unknown → tag as "Neutral Impact".
- If model inference fails → fallback to rule-based baseline.

---

3. Frontend Layer

3.1 Framework and UI Overview

The EC module provides an interactive calendar integrated into the Goldh front-end shell. Users can browse upcoming events, visualize AI impact predictions, and correlate macro data with asset trends.

Key UI Components:

- **Event Calendar:** sortable by region, category, or importance.
- **Impact View:** displays predicted vs actual market reactions.
- **Sentiment Overlay:** integrates GID and UMF confidence scores.

3.2 MVP Implementation

- Static pages refreshed hourly from cached event JSON.
- Simple color-coded importance scale (High, Medium, Low).
- REST-based backend queries.
- Next.js + Recharts used for calendar visualization.

3.3 Phase 2 Implementation

- Real-time WebSocket updates for new events.
- Impact animation with AI confidence bands (0–100).
- Integration with UMF dashboard via shared GraphQL API.
- Personalized event filtering by user preferences and holdings.

3.4 Cross-Module References

- **UMF TDD 3.4:** shares frontend component for impact heatmap rendering.
- **GID TDD 3.4:** uses EC macro sentiment context in narrative panels.

- **TAC TDD 3.2:** analyst update screens include macro impact annotations.

4. Backend Layer

4.1 Overview

The backend orchestrates all macroeconomic data ingestion, event normalization, and AI impact scoring workflows.
It's built as a set of stateless, horizontally scalable microservices for reliability and modularity.

Core Services

- **Event Ingestion Service** – pulls or listens for event updates from providers.
- **Event Processor** – cleans, deduplicates, and timestamps macro data.
- **AI Impact Service** – calculates short-term and long-term impact metrics.
- **Distribution Service** – sends enriched results to UMF, TAC, and GID.

4.2 Responsibilities

- Normalize country and currency codes.
- Parse event attributes (importance, forecast, actual, previous).
- Run impact models and return normalized impact scores.
- Publish enriched results to internal APIs and queues.
- Handle retry, logging, and monitoring.

4.3 MVP Implementation

- Poll-based ingestion using AWS Lambda cron jobs.
- JSON normalization pipelines implemented in Python.
- Impact engine: rule-based mapping of "Surprise %" to impact level.
- Output cached in Redis; daily snapshots stored in S3.
- Basic alerting through CloudWatch.

4.4 Phase 2 Implementation

- Event-driven design using Kinesis streams for real-time updates.
- Impact engine migrated to SageMaker inference endpoints.
- Model registry with version control and drift detection.
- Microservices containerized on ECS with auto-scaling.
- Full observability via Datadog dashboards and OpenTelemetry traces.

4.5 Cross-References

- **UMF TDD 4.3 – 9.4:** EC event data consumed by UMF analytics pipeline.
- **GID TDD 9.4:** EC streaming schema reused for sentiment correlation.

---

5. API Gateway & Integration

5.1 Overview

The EC API Gateway manages all inbound and outbound connections to ensure secure, throttled, and observable integrations.
It exposes both REST and GraphQL interfaces for downstream modules.

5.2 MVP Integration Model

- **Type:** Pull model (HTTP GET polling).
- **Providers:** EconoData, Investing.com, Alpha Vantage.
- **Auth:** API Key or Token.
- **Frequency:** 15 – 60 min intervals.
- **Schema Validation:** OpenAPI 3.1.
- **Error Handling:** Retries with exponential back-off.
- **Cache:** Redis TTL (1 hour default).

5.3 Phase 2 Integration Model

- **Type:** Streaming / Webhooks from paid providers.
- **Providers:** Trading Economics, Refinitiv, Benzinga Calendar.
- **Auth:** OAuth2 + mTLS.
- **Schema Registry:** Avro stored in S3.
- **Event Bus:** Kinesis / SNS for distribution.
- **Observability:** Latency and error budget metrics exported to Datadog.
- **Audit:** Signed payloads and sequence IDs for replay.

5.4 Integration Flow

1. External API → Gateway → Event Processor.
2. Processor validates schema and normalizes payload.
3. AI Impact Service computes impact score.
4. Results pushed to Redis and Feature Store.
5. Distribution Service publishes to Confidence Bus and UMF analytics.

5.5 Cross-References

- **UMF TDD 5.4:** shared gateway flow pattern.
- **TAC TDD 5.1:** confidence service contract used by EC integration.

---

6. Data Layer & Storage

6.1 Overview

The data layer manages event storage and historical impact data for backtesting and AI training.
It supports both hot cache and cold archive tiers for read efficiency and long-term durability.

6.2 MVP Design

- **Cache:** Redis with 60 min TTL for event summaries.
- **Cold Store:** S3 Parquet files partitioned by country and date.
- **ETL:** Daily batch for cleanup and compaction.
- **Retention:** 180 days.
- **Schema:** JSON (flat) → Parquet (columnar).

6.3 Phase 2 Design

- **Primary DB:** PostgreSQL (Aurora RDS) partitioned by region and event type.
- **Analytics Store:** Redshift for cross-event impact correlation.
- **Feature Store:** SageMaker Feature Store for training ML impact models.
- **Archive:** Glacier Deep Archive (7 years).
- **Indexing:** B-tree on (event_id, region, impact_score).
- **Replication:** Cross-region replica for DR.

6.4 Data Governance & Security

- KMS encryption at rest; TLS 1.3 in transit.
- IAM least-privilege roles per service.
- GDPR and Swiss PII compliance for user preferences.
- Automated retention policy enforced via Lifecycle Rules.

6.5 Cross-References

- **UMF TDD 6.5:** shared data retention strategy.
- **GID TDD 6.5:** parallel governance model for insider feeds.

7. AI Impact Model Subsystem

7.1 Overview

The **AI Impact Model Subsystem** quantifies the potential market impact of macroeconomic and earnings events.
It evaluates historical relationships between event surprise values, sentiment shifts, and asset price movements.

7.2 Core Functions

1. **Event Feature Engineering:** Extracts macro factors such as forecast deviation, historical volatility, and cross-asset correlation.
2. **Impact Prediction:** Calculates short-term (24h) and long-term (7d) impact scores.
3. **Confidence Band Generation:** Produces confidence intervals for AI impact predictions.
4. **Explainability:** Uses SHAP-based interpretation for human validation.
5. **Feedback Loop:** Monitors accuracy and retrains on new events weekly.

7.3 MVP Implementation

- Deterministic mapping of forecast deviation (%) → impact level (−2 to +2).
- Rule-based classification by event category (Inflation, Employment, Crypto Regulation).
- Batch jobs executed on AWS Lambda daily.
- Historical calibration using CSV datasets.
- Simple MAE error tracking, manual retraining.

7.4 Phase 2 Implementation

- Regression-based models trained in SageMaker (XGBoost, LSTM).
- Continuous online inference for real-time event updates.
- Multi-feature vector: includes GID sentiment and TAC consensus deltas.
- Retraining triggered by drift detection via Model Monitor.
- Explainable AI output for impact rationale visualization in dashboards.

7.5 Cross-References

- **UMF TDD 7.5:** shared AI feature pipeline and scoring model integration.
- **TAC TDD 7.2:** macro factor weighting used for analyst confidence blending.

---

8. Confidence Score Adjustment Engine

8.1 Purpose

The Confidence Score Adjustment Engine modifies the baseline market confidence (shared across UMF and TAC) based on macroeconomic events and their AI-predicted impact.

8.2 Inputs

- Event importance level (High / Medium / Low).
- Forecast vs actual deviation (Surprise %).
- Event category (e.g., CPI, GDP, Non-Farm Payrolls).
- AI-predicted market impact (from 7).
- Cross-module sentiment score from GID.

8.3 MVP Model

- Static weight adjustment: Confidence Δ = Base × (Impact Factor × Weight).
- Update schedule: hourly batch.
- Stored in Redis; fallback to neutral adjustment if missing data.
- Manual override allowed for key events (FOMC, ECB).

8.4 Phase 2 Model

- Multi-variable regression model integrating sentiment and consensus deltas.
- Outputs adjustment coefficients per sector and asset type.
- Real-time application through event stream triggers.
- Confidence Bus broadcasts updates system-wide (<250 ms latency).
- Drift analysis for model retraining.

8.5 Cross-References

- **UMF TDD 8.6:** defines integration for global confidence scoring.
- **GID TDD 8.6:** sentiment weight fusion for event impact propagation.

---

9. Event Pipeline (Async & Streaming)

9.1 Overview

The Event Pipeline delivers live macroeconomic updates and impact scores to consuming modules.
It forms the backbone for near-real-time event broadcasting across the Goldh ecosystem.

9.2 MVP Implementation

- Batch job queue managed by AWS SQS.
- 15-minute polling frequency.
- Dead-letter queue for unprocessed or invalid events.
- Lambda-triggered ETL process to clean and requeue data.
- Basic CloudWatch logging and retries.

9.3 Phase 2 Implementation

- Kinesis Data Streams for continuous event flow.
- Parallel consumers for UMF, GID, and TAC.
- Schema registry for version management.
- Replay support for impact analysis and backtesting.
- Real-time metrics (P95 latency < 200 ms).

9.4 Cross-References

- **UMF TDD 9.4:** shared asynchronous event taxonomy.
- **GID TDD 9.4:** parallel schema registry for sentiment event propagation.

10. Service Mesh & Inter-Module Communication

10.1 Overview

The EC module participates in the Goldh service mesh for secure, observable, and resilient inter-module communication.
It interacts primarily with UMF, TAC, and GID, sharing real-time event impact updates and macro context.

10.2 MVP Design

- RESTful calls within the same AWS VPC.
- Static service registry (Parameter Store).
- JWT-based inter-service authentication.
- Retry logic limited to 3 attempts per failure.
- Metrics collected via CloudWatch and SNS alerts.

10.3 Phase 2 Design

- Adoption of **AWS App Mesh** or Istio for mutual TLS (mTLS) and dynamic discovery.
- SLO-based traffic routing and retries.
- Circuit breaker patterns for resiliency.
- OpenTelemetry tracing propagated across UMF, TAC, and GID calls.
- Unified metrics export to Datadog for cross-module visibility.

10.4 Cross-References

- **UMF TDD 10.4:** shared service mesh policy.
- **GID TDD 10.4:** sentiment streaming dependencies.
- **TAC TDD 10.4:** macro impact feed integration for analyst view enrichment.

---

11. CI/CD and Deployment

11.1 Overview

Continuous Integration and Deployment (CI/CD) pipelines automate EC builds, testing, and releases while ensuring compliance and auditability.

11.2 CI Pipeline

MVP:

- GitHub Actions or CodeBuild runs for every commit.
- Unit and schema validation tests.
- Basic static analysis using Bandit.
- Manual artifact tagging and promotion.

Phase 2:

- Multi-stage pipeline: Unit → Integration → Contract → E2E tests.
- Model validation step (Impact Engine drift detection).
- Signed container images (Sigstore / Cosign).
- Supply-chain scanning integrated with AWS Security Hub.

11.3 CD Pipeline

MVP:

- Manual staging to production promotion.
- Lambda or Fargate deployment targets.
- Rollback triggered manually through AWS CLI.

Phase 2:

- Canary and Blue/Green releases via CodeDeploy.
- Automatic rollback on failed SLO verification.
- Infrastructure as Code (Terraform) with environment tagging.
- Multi-region deployment for HA and low latency.

11.4 Cross-References

- **UMF TDD 11.4:** shared CI/CD workflow templates.
- **GID TDD 11.4:** NLP model deployment consistency.
- **TAC TDD 11.4:** analyst consensus service integrated testing pipeline.

---

12. Performance, Scalability & Observability

12.1 Overview

The EC module is designed to meet real-time analytics requirements for macro event ingestion and AI inference while maintaining high availability and traceability.

12.2 Performance Targets

Metric	MVP	Phase 2
Event Ingestion Latency	< 1 min	< 10 s
Impact Model Inference Latency	< 2 s	< 300 ms
Dashboard Update Frequency	Hourly	Real-time
Uptime	99.5%	99.9%

12.3 Scalability Patterns

- **MVP:** Lambda concurrency scaling, Redis caching, and asynchronous polling.
- **Phase 2:** Kinesis stream scaling via shards; ECS task autoscaling; Aurora read replicas for load balancing.

12.4 Observability Stack

- **MVP:** CloudWatch dashboards for ingestion and model latency metrics.
- **Phase 2:** Unified Datadog dashboards, Prometheus exporters, OpenTelemetry traces.
- Synthetic probes for uptime verification.
- Event anomaly detection integrated with SageMaker Model Monitor.

12.5 Cross-References

- **UMF TDD 12.6:** shared telemetry schema and SLO definitions.
- **GID TDD 12.5:** event-level observability integration.
- **TAC TDD 12.5:** analyst feed latency coordination.

13. Security & Compliance (Swiss DLT + GDPR)

13.1 Overview

The EC module ingests and enriches public economic data, but user-specific preferences and regional filters require strict data-protection alignment with Swiss FINMA DLT guidelines and GDPR.

13.2 Core Principles

- **Data Minimization:** Store only identifiers, not full PII.
- **Encryption:** TLS 1.3 in transit and AES-256 at rest.
- **Access Control:** IAM roles with least privilege.
- **Auditability:** Immutable CloudTrail logs and S3 object versioning.
- **Compliance:** GDPR Articles 5–32; Swiss Data Protection Act.
- **Incident Response:** Alert within 24 h of breach to compliance officer.

13.3 MVP Implementation

- Parameter Store for secret management.
- JWT auth between frontend and BFF.
- Manual audit exports for regulatory review.
- Consent captured at signup and stored in user profile service.

13.4 Phase 2 Implementation

- AWS Secrets Manager and certificate-based auth for inter-service calls.
- Automated compliance checks embedded in CI/CD.
- Consent revocation API for end-users.
- DLP (Data Loss Prevention) scanner for ingested free-text data.
- Cross-region PII tokenization for EU/CH data sovereignty.

13.5 Cross-References

- **UMF TDD 13.5:** centralized security policies.
- **GID TDD 13.5:** shared audit and erasure workflow.
- **TAC TDD 13.3:** analyst data governance.

---

14. Testing & QA Framework

14.1 Testing Strategy

A multi-layer testing pyramid ensures event accuracy and impact-model integrity.

Layer	Objective	Tools	Frequency
Unit	Validate ingestion and parsing logic	PyTest, Jest	Every commit
Contract	Verify API schemas	Dredd, Postman	Every build
Integration	Ensure module inter-operation	k6, Docker Compose	Nightly
End-to-End	Validate macro → confidence flow	Cypress	Pre-release
AI/ML	Validate impact prediction accuracy	SageMaker Monitor	Weekly

14.2 MVP Scope

- Unit and contract tests required for all commits.
- Mocked provider APIs for EconoData and Alpha Vantage.
- Manual visual check of calendar output in staging.

14.3 Phase 2 Scope

- Automated E2E tests across UMF/GID/TAC flows.
- Performance and load testing under concurrent updates.
- Regression tests for ML model retraining.
- Chaos tests for event-bus resilience.
- Security and drift scans integrated with CI/CD.

14.4 QA Environments

- Dedicated staging cluster mirrors production.
- Sandbox API keys for paid provider integration.
- Synthetic event dataset for test automation.

14.5 Cross-References

- **UMF TDD 14.5:** shared confidence regression suite.
- **GID TDD 14.5:** sentiment pipeline test alignment.
- **TAC TDD 14.5:** analyst impact correlation tests.

---

15. Phase Differentiation Matrix (MVP vs Phase 2)

Component	MVP Implementation	Phase 2 Implementation
Data Sources	EconoData, Alpha Vantage	Refinitiv, Trading Economics, Benzinga
Processing	Rule-based impact mapping	AI regression models with confidence bands
Storage	Redis + S3	PostgreSQL + Redshift + Feature Store
AI / ML	Static rules	SageMaker real-time inference + XAI
Security	JWT + TLS	mTLS + IAM + automated compliance
Delivery	Hourly batch updates	Real-time streams to UMF / TAC
Testing	Unit and manual QA	Automated regression + load tests

16. Cross-Module Dependency Summary

16.1 Overview

The EC module serves as a foundational macro-context layer for the Goldh ecosystem. It feeds event-based sentiment and impact data into the UMF, GID, and TAC modules for a unified market intelligence experience.

Module	Dependency Type	Reference	Integration Mechanism
UMF (Universal Market Financials)	Macro event data + AI impact score	UMF TDD 6.5 / 9.4 / 12.6	Real-time Confidence Bus and REST aggregation
GID (Guru & Insider Digest)	Contextual macro sentiment	GID TDD 9.4 / 12.5	Shared streaming schema for event enrichment
TAC (Top Analysts Consensus)	Event-driven confidence modifier	TAC TDD 7.2 / 12.5	Macro impact fusion in confidence aggregation service

16.2 Data Exchange Model

- **Transport:** HTTPS and mTLS inside the service mesh.
- **Format:** JSON and Avro contracts in central schema registry.
- **Versioning:** v1 (MVP) → v2 (Phase 2) for event model evolution.
- **Error Handling:** Circuit breakers, retries, DLQ for failed updates.

16.3 Shared AI / Feature Infrastructure

- Feature Store shared with UMF for volatility and macro features.
- Model registry shared across EC + TAC for confidence calibration.
- Periodic retraining pipeline using consolidated macro + sentiment datasets.

16.4 Security & Compliance Dependencies

- Common KMS keyspace across EC, UMF, and GID.
- Unified audit log collector for GDPR Article 30 accountability.
- Centralized erasure workflow for user preferences (via event bus).

17. References & Glossary

17.1 Primary References

- **EC BRD v1.0** – Business Requirements Document
- **Goldh API Master List Workbook (2025)**
- **UMF TDD v1.0** – Universal Market Financials Design
- **GID TDD v1.0** – Guru & Insider Digest Architecture
- **TAC TDD v1.0** – Top Analysts Consensus Design

17.2 External References

- FINMA DLT Circular 04/2024
- GDPR Articles 5–32 (EU Regulation 2016/679)
- AWS Architecture for Real-Time Analytics (2025 Edition)
- spaCy / FinBERT / AWS SageMaker Documentation

17.3 Glossary

Term	Definition
EC	Economic Calendar module providing macro and earnings event data
Confidence Bus	Cross-module event stream for confidence and impact propagation
Feature Store	Centralized ML feature repository for training and inference
Impact Score	0–100 scale quantifying expected market response to events
DLQ	Dead Letter Queue for failed event delivery
App Mesh	AWS service mesh enabling mTLS and traffic management
Phase 2	Premium release with real-time feeds and AI impact forecasting
MVP	Minimum Viable Product – free tier using open macro feeds