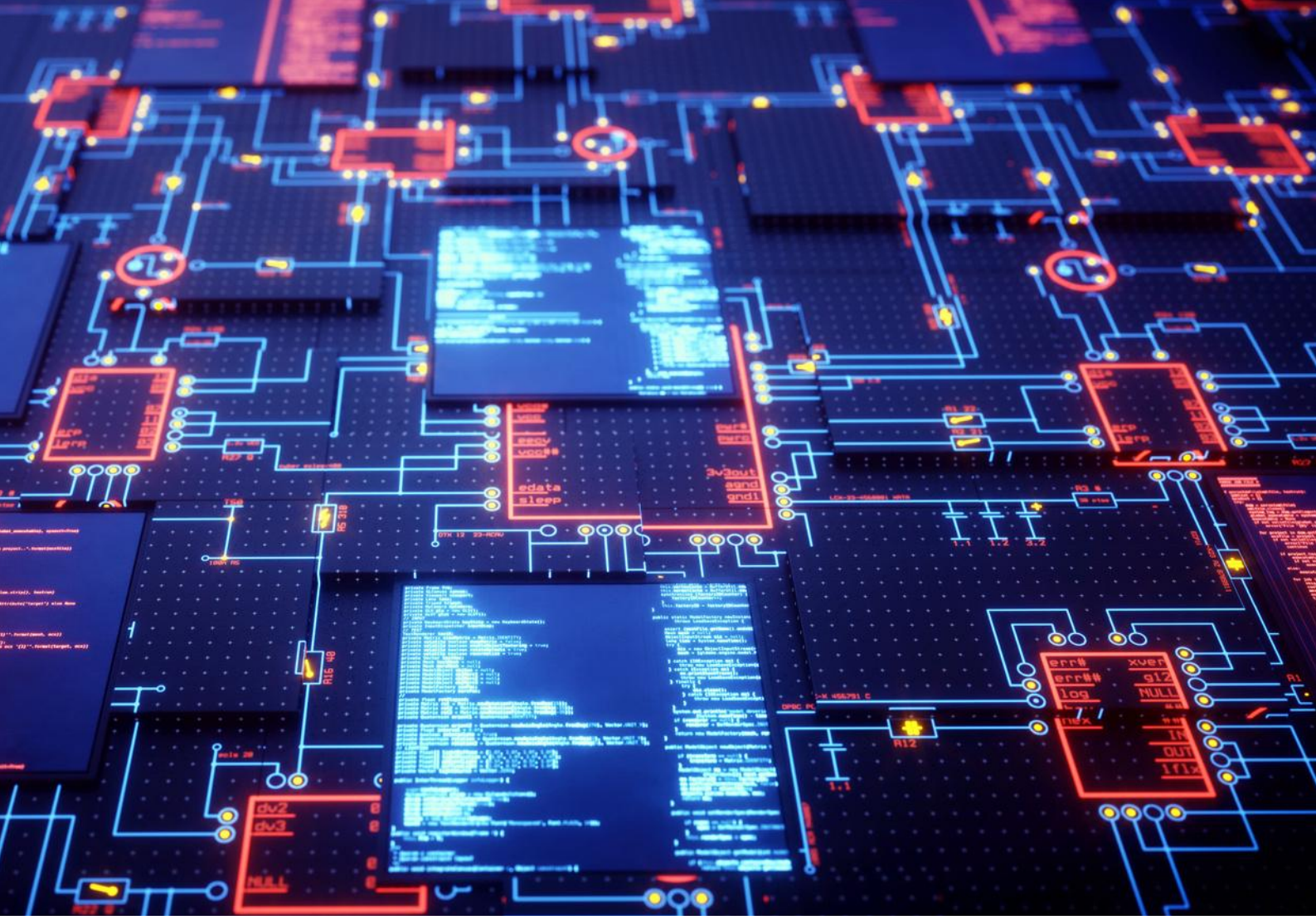


WHITE PAPER

Integrating Post-Trade FIX Drop Copy & Trade Capture Workflows with Modern Cloud Data Platforms (Databricks, Snowflake)

This paper addresses a central architectural challenge in modern trading infrastructure: **how to reliably ingest, process, and store high-volume post-trade execution data from FIX drop copy and trade capture sessions into cloud data platforms (e.g., Databricks, Snowflake).**

These capabilities are required for **analytics, risk management, algorithmic TCA, surveillance, compliance, reconciliation, and machine learning.**

Background:

- **FIX Drop Copy** is a mechanism (using the FIX protocol) by which a broker, trading desk, or exchange sends copies of order-flow or execution messages downstream to another system (e.g. to a risk, compliance, or clearing system). Essentially, it's like a "carbon copy" of trading messages (orders, cancels, fills) to a secondary consumer. Example: Drop Copy solution
- **Trade Capture (Feed)** is similar, more focused on fills/execution reports (i.e. actual trade confirmations) rather than the full life of the order. It may be delivered via FIX (or a variant) or via other protocols. Example: Trade Capture solution

Assumptions & Goals

To ground things:

- You receive one or more FIX drop copy / trade capture sessions from counterparties or exchanges.
- You want near-real-time ingestion (latency target: e.g. 1-5 seconds) to support dashboards, monitoring, risk, compliance, and ML.
- You also want high throughput, resilience, replayability, and the ability to reprocess (backfill).
- You will host FIX server on the cloud, establishing outbound connections to the venues
- You will use Databricks (Spark / Delta Lake) as your streaming/transformation engine, and Snowflake as your analytical / BI serving layer.
- You expect evolving schemas, occasional outages, and the need for stateful joins / enrichment / deduplication.

Challenges in the Integration (functional)

Here are some of the key challenges to address with post- trade stream integrations :

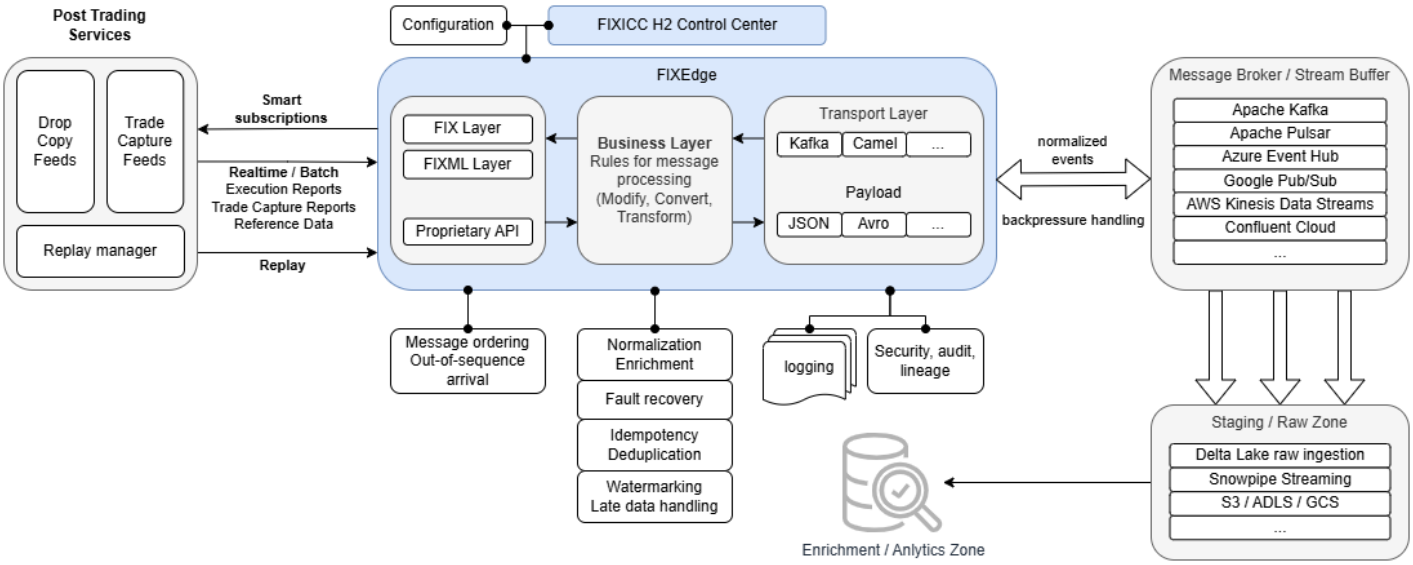
Challenge	Description & Implications
Message ordering / out-of-sequence arrival	FIX messages may not arrive strictly in sequential order. Some messages (or partial) may be delayed or delivered out-of-order. You need logic to re-sequence or handle late-arriving updates.
Idempotency & deduplication	FIX feeds may occasionally resend or retransmit some messages. Your pipeline must dedupe or filter duplicates via unique keys (e.g. message sequence number or trade ID).
Schema evolution / versioning	Over time, new FIX tags / message types might be added, or existing fields may change. Your pipeline must accommodate schema evolution
Raw and curated data storage	You need both raw FIX message store (for audit / replay), and curated trade / order tables for BI / risk / TCA.
Normalization / enrichment	Raw FIX messages often need normalization (mapping tags to domain names), enrichment (e.g. instrument metadata, reference data join), and normalization across multiple feed sources.
Data Lifecycle & Replay	Many systems do not distinguish between original execution and post-replay corrected events, requiring capture of “effective time” vs “processing time”.
ID consistency	Mapping ClOrdID, OrigClOrdID, ExecID, TradeID, AllocID, etc. into stable, unique trade/order IDs for analytics.
Transactional consistency / atomicity	For any given trade, you may have multiple FIX messages (e.g. order, fills, cancels) that need to be joined or correlated. Ensuring atomicity in how these land in your warehouse is nontrivial.
End-to-End Traceability	From FIX ExecID through middleware, into Databricks/Snowflake row. You need consistent correlation IDs and logs across FIX server, Middleware, Warehouse.
Schema design & partitioning	How you partition (e.g. by date, instrument, account) in your target tables strongly affects performance and cost.

Challenges in the Integration (non-functional)

Challenge	Description & Implications
Integration with the middleware pattern	Synchronize, asyncronize with queues, backpressure mechanism, transactional boundaries, error handling.
Throughput / scale	In a high-frequency trading environment, you may see extremely high message rates; the system must scale.
Fault recovery	In the event of a connection drop, you must be able to request missing messages (recover) or replay historical data to ensure no gaps.
Latency vs batch tradeoffs	Real-time ingestion is desired, but sometimes you may adopt micro-batches to balance cost vs latency.
Watermarking / late data handling	Especially in streaming, you need to define how much lateness you can tolerate, and how to handle very late-arriving messages.
Security, audit, lineage	You need strong security, comprehensive audit trails, and full data lineage across the pipeline.
Cloud deployment	The ports used for establishing connections must be allowed in the security groups within the VPC. Firewall and NAT rules must allow access to the venue hosts.
High availability	High Availability can be ensured through Kubernetes cluster mechanisms . For the state management of FIX sessions, shared storage accessible to all nodes (this can be Azure Files, AWS EFS, or a similar solution) could be leveraged.

Refer to Figure 1, for the end-to-end FIX protocol post-trading processing architecture using EPAM solution

Figure 1



The main workflow path

Ingestion layer

1. FIX Engine / FIX Drop Copy receiver

The production-grade FIX server FIXEdge to connect to the drop copy / trade capture feed. FIXEdge provides solutions that handle recovery, connectivity, and session management.

The following Post Trade Message Types supported:

- Security Definition Request
- Security Definition Response
- User-Defined Strategy
- Trade Capture Report Request
- Trade Capture Report Request Ack
- Trade Capture Report
- Allocation Report
- News
- Execution Report
- History Request
- Order Cancel Reject
- Order Mass Cancel Request
- Order Mass Cancel Report
- Quote Acknowledgement

2. The receiver (FIXEdge) then unmarshals the FIX messages into a structured, canonical format (e.g. JSON, Avro, protobuf, or relational rows).

3. Message Broker / Stream Buffer

FIXEdge pushes the normalized events into a streaming buffer (e.g. Apache Kafka, Apache Pulsar, Azure Event Hubs, AWS Kinesis Data Streams, Google Pub/Sub). This decouples the FIX ingestion from downstream transformation. It also gives you benefits in replay, backpressure handling, and fan-out.

4. Staging / Raw Zone in Lakehouse / Data Warehouse

From the streaming buffer, ingest into a raw "staging" layer, often using a streaming ingestion engine (Spark Structured Streaming, Flink, etc.) or via connectors (e.g. Kafka → Iceberg / Delta / Snowflake streams).

- For Databricks: you might land directly into Delta Lake tables (raw zone).
- For Snowflake: you might use Snowpipe (continuous ingestion) or streams + tasks.
- Alternatively, you could stage raw JSON / Avro files in cloud object storage (S3 / ADLS / GCS) and then load.

Transformation / Enrichment Layer

Once in staging, you run transformations to normalize, join with reference data (e.g. instrument master data, account metadata), deduplicate, reorder (if necessary), and generate final "analytics" tables.

- In Databricks, you might use Spark jobs, Delta Live Tables (or Lakeflow declarative pipelines), or structured streaming + stateful operators.
- In Snowflake, you might use Snowflake Streams + Tasks, or ELT pipelines (e.g. leveraging DBT or stored procedures) to process the staging changes into transformed tables.

You also need to handle **Change Data Capture (CDC)** semantics – e.g. updates (cancels, corrections), deletes (trade cancellations), new inserts, etc.

Databricks has recently introduced **AUTO CDC APIs** in Lakeflow Declarative Pipelines to simplify handling of change data feeds, automatically managing out-of-order records. (docs.databricks.com) That helps with streaming update logic in a correct, robust way.

Serving / Analytics Layer

Finally, the processed, normalized tables are exposed to BI tools, dashboards, machine learning pipelines, or downstream risk / compliance systems.

- In a Databricks + Lakehouse architecture, you might keep your analytical tables in Delta format (managed, optimized) and allow query engines (Spark SQL, SQL endpoints) to serve them.
- In a Snowflake-based architecture, your transformed tables live inside Snowflake. Upstream pipelines (e.g. Snowpipe, Streams, Tasks) feed them, and users query via SQL, BI tools, etc.

EPAM Solution Accelerator

EPAM's FIXEdge Trade Capture and Drop Copy modules, combined with our Cloud Data Engineering Framework (Databricks / Snowflake connectors and streaming enrichment templates), help financial institutions address challenges discussed above and accelerate implementation of this architecture by providing:

- Pre-configured FIX setup for trading venues and brokers
- Structured message parsing and canonicalization
- Built-in session recovery and replay handling logic
- Quick-start integration blueprints for Kafka, Event Hubs, and cloud ingestion pipelines
- Proven adapters for Delta Lake, Snowflake Streams & Tasks, and Databricks structured streaming
- Ready for cloud deployment. It can be deployed as a standalone Virtual Machine using standard RPM/DEB packages or Docker images and orchestrated within a Kubernetes cluster. Additionally, FIXEdge is available as a pre-configured offering in major cloud marketplaces.
- Regulatory-compliant security model. TLS connection is used for securing communication between all components of the solution, as well as for connections to the venue. The solution can be integrated with various KeyVault systems for storing sensitive data that support REST API (e.g., Azure Key Vault, AWS Secrets Manager, HashiCorp Vault).
- Observability & Automation. The FIXEdge has a built-in REST API server that allows monitoring the state of the application and all its sessions, as well as managing the application. This enables the integration of FIXEdge with numerous Cloud Services and the automation of many processes (i.e. CloudWatch, Azure Monitor).
- Reconciliation tooling. Integrated solution comes with Auto recon jobs comparing FIXEdge store vs DW facts by day.

This reduces implementation time from months to weeks and significantly mitigates delivery and operational risks.

Conclusion

The integration of FIXEdge post-trade workflows with cloud-native analytics platforms such as Databricks and Snowflake marks a strategic transition from traditional point-to-point post-trade infrastructure toward a scalable, resilient, and insight-driven architecture. While challenges exist around message ordering, lifecycle alignment, replay handling, schema evolution, multi-hops across middleware, and regulatory compliance, these can be addressed through:

- Well-defined ingestion architecture with idempotent processing
- Canonical trade data models
- Robust replay strategies
- Secure and governed cloud deployment

By leveraging EPAM's B2BITS FIXEdge trade capture modules and our cloud data engineering accelerators, firms can:

- Reduce time-to-market for advanced analytics and compliance reporting
- Lower the cost and risk associated with custom development
- Improve operational reliability and observability
- Establish a foundation for real-time risk monitoring, TCA, and AI-driven decision engines

Organizations that implement this architecture early will be better equipped to provide real-time post-trade transparency and respond rapidly to market and regulatory change.

For more information about our products, visit:
www.b2bits.com
Solutions Overview

Contact us: **sales@btobits.com**