# Draft Proposal: Batch Inference API for FX Deals on Databricks

Date: 03 Oct 2025

## 1) Objective (Updated)

Shift from table-only batch to an API-first design. External teams will submit a batch for inference via an API. Each submission includes a batch\_id and a list of deal identifiers. The system processes the batch asynchronously, writes outcomes to a Delta table, and provides a simple way for the caller to know when processing is complete (listener/status endpoint or polling with an ETA).

## 2) High-Level Approach

• External app calls a Submit API with batch\_id + deal\_ids (+ optional metadata).  
• System enqueues the work by writing rows with status NEW into a Delta table.  
• A Databricks Workflow (scheduled or event-driven) picks NEW items, runs inference using infer.py in batch.  
• Results are written back to the same table; per-row status moves to DONE/ERROR.  
• Caller learns completion via either:  
 – (A) Listener/Status API to check batch status, or  
 – (B) Polling the table-backed status endpoint after an estimated time window.

## 3) API Design (Simple & Practical)

### 3.1 Submit Batch

POST /v1/batches

Request (JSON):

{  
 "batch\_id": "2025-10-03-ACME-42",  
 "deal\_ids": ["D123","D124","D125"],  
 "source\_system": "ACME-OMS",  
 "priority": 5,  
 "callback\_url": "https://client.app/webhooks/fx-batch"  
}

Response (JSON):

{  
 "batch\_id": "2025-10-03-ACME-42",  
 "accepted": true,  
 "estimated\_ready\_in\_seconds": 600  
}

Behavior:  
• Server validates batch\_id uniqueness; idempotent on repeat submissions.  
• Writes one row per deal to the Delta queue table with status NEW and the provided batch\_id.  
• If callback\_url is present, a completion notification will be attempted when all rows finalize.

### 3.2 Get Batch Status (Listener/Status Endpoint)

GET /v1/batches/{batch\_id}

Response (JSON):

{  
 "batch\_id": "2025-10-03-ACME-42",  
 "summary": {  
 "total": 3,  
 "done": 3,  
 "error": 0,  
 "processing": 0,  
 "new": 0  
 },  
 "status": "COMPLETED",  
 "completed\_at": "2025-10-03T10:12:00Z"  
}

Behavior:  
• Aggregates per-row statuses from the Delta table.  
• Marks the batch COMPLETED when all rows are DONE; FAILED if any hard failure stops the run; PARTIAL if some are ERROR but the job finished.

### 3.3 Optional: Webhook Callback

If the client supplies callback\_url on submission, the system will POST a completion payload at the end of processing:  
{  
 "batch\_id": "2025-10-03-ACME-42",  
 "status": "COMPLETED",  
 "summary": { "total": 3, "done": 3, "error": 0 },  
 "link": "https://our.api/v1/batches/2025-10-03-ACME-42"  
}  
Retries with exponential backoff if client endpoint is temporarily unavailable.

## 4) Data Model (Delta Tables)

Queue/Result (single-table pattern): fx\_deals\_inference\_queue

Columns:  
• deal\_id (STRING)  
• batch\_id (STRING)  
• payload (STRING)  
• status (STRING) // NEW, PROCESSING, DONE, ERROR  
• result (STRING)  
• error\_message (STRING)  
• model\_version (STRING)  
• submitted\_at (TIMESTAMP)  
• processed\_at (TIMESTAMP)

## 5) Processing Flow (Simple)

1. Submit API writes one row per deal with status NEW and batch\_id.  
2. Databricks job wakes up (schedule or trigger), claims NEW rows → PROCESSING.  
3. Batched inference via infer.py (load model once; process DataFrame).  
4. Write results per row; set DONE or ERROR; stamp model\_version and processed\_at.  
5. If all rows in a batch are finalized, batch becomes COMPLETED; optional webhook fired.  
6. Status endpoint surfaces real-time progress using the view.

## 6) How the Client Knows It’s Done

Option A — Listener/Status API: Client calls GET /v1/batches/{batch\_id} until status is COMPLETED or FAILED.  
Option B — Webhook: We call client’s callback\_url once finished (with retries).  
If neither is feasible, we return estimated\_ready\_in\_seconds at submission time so the client can check after that window.

## 7) SLA & Estimated Completion Time

We will compute a simple ETA at submission based on recent median throughput:  
ETA(seconds) ≈ ceil( (queue\_backlog + batch\_size) / deals\_per\_second\_recent )  
  
Initial guidance (tunable by environment):  
• Small batches (≤1k deals): ~20 minutes  
• Medium (1k–10k): 60 minutes  
• Large (10k+): depends on cluster size; autoscaling recommended

## 8) Open Decisions

• Maximum batch size per submission?  
• Required SLA and timeout behavior?  
• Retention policy for raw payload vs. summarized results?  
• Do we need per-tenant throttling/priority?

Summary: Provide a simple Batch Submit API that writes to a Delta-backed queue, process asynchronously on Databricks, and expose a Status API (and optional webhook). Results remain in the table for audit and downstream use.