NETCELL EDA PLATFORM

Event-Driven AI Platform Design

NETCELL-EDA Architecture

Platform Design

1. Model registry design – part1

Goal: Central place to store, version, and manage all AI models (risk, churn, engagement, etc.) so Inference and Profiling Services can use them safely.

Core concepts

Model: a single trained artifact (file).

Model Type: e.g. Risk, Churn, Engagement.

Version: e.g. risk_v3, churn_v2.

Status: draft, validated, production, deprecated.

Tenant scope: global or per TenantId.

Storage

Metadata table (SQL / Hive):

Model files stored in:

S3 / HDFS / Cloudera storage.

API

GET /Api/models/active?modelType=Risk&tenantId=tenantA → returns active model metadata + path.

POST /api/models → register new model version.

PUT /api/models/{modelId}/status → change status to production.

NETCELL-EDA Architecture

Platform Design

1. Model registry design –part2

Usage

Inference Service:

On startup: loads active models from registry.

On config change: reloads models.

Training pipeline:

After training: registers new model with metrics.

Ops decides when to promote to production.

NETCELL-EDA Architecture

Platform Design

2. Profiling service internals - part1

Goal: Take events + features, apply business logic + AI, and update entity profiles (Contact, Account, etc.).

Responsibilities

Build feature vectors.

Call Inference Service.

Merge scores into profiles.

Apply rules (e.g. thresholds, flags).

Write profiles to Document DB.

Internal components

Profile Loader

Reads profile from Document DB by TenantId + EntityId.

Feature Builder

Uses profile + latest events to compute features.

Writes features to Online Feature Store.

NETCELL-EDA Architecture

Platform Design

2. Profiling service internals-part2

Inference Client

Calls Inference Service with features.

Receives scores.

Profile Updater

Updates:

State

StateHistory

Scores

LastEventAt

Upserts profile.

Rules Engine (optional)

Applies business rules:

if Risk > 0.8 → flag HighRisk.

if Churn > 0.7 → mark AtRisk.

NETCELL-EDA Architecture

Platform Design

3. Event processor code flow

Goal: Define how a single microservice processes events from Kafka.

Example: Contact Events Processor

High-level flow:

Consume event from contacts-events topic.

Validate & deserialize using schema registry.

Call Profiling Service (or internal profiling logic).

Update profile in Document DB.

Optionally emit new events (e.g. ContactRiskUpdated).

Pseudo-code:

```
loop:
  event = kafkaConsumer.poll()
 if event.EventType == "ContactStateChanged":
      profile = ProfileLoader.load(event.TenantId, event.Data.ContactId)
 features = FeatureBuilder.build(profile, event)
 scores = InferenceClient.scoreContact(event.TenantId, event.Data.ContactId, features)
 updatedProfile = ProfileUpdater.update(profile, event, scores)
 DocumentDb.upsert(updatedProfile)
```

NETCELL-EDA Architecture

Platform Design

4. Kafka consumer group strategy

Goal: Scale event processing safely and predictably.

Key ideas

One consumer group per microservice per topic.

- contact-processor-group for contacts-events.
- billing-processor-group for billing-events.

Parallelism via partitions.

- If contacts-events has 8 partitions:
- You can run up to 8 consumer instances in contact-processor-group.
- Kafka will assign partitions to instances.

Ordering guarantee.

- Using ContactId as key ensures all events for one contact go to the same partition → processed in order.

Strategy

Start with:

- 1 consumer group per domain processor.
- 4–8 partitions per topic.
- Scale by:
- increasing instances in the consumer group.
- increasing partitions if needed.

NETCELL-EDA Architecture

Platform Design

5. Tenant isolation in microservices

Goal: Ensure tenants are logically isolated, even if they share topics and services.

Approaches

TenantId in every event.

Mandatory field.

Used for filtering, routing, and access control.

Tenant-aware processing.

Event processors:

read TenantId

apply tenant-specific config (thresholds, models, rules).

Config per tenant.

Central Configuration Service:

GET /config/tenantA/contact → returns rules, thresholds, model types.

Security:

API Gateway enforces:

which tenant can access which dashboards.

Cloudera Ranger enforces:

which tenant can access which data lake tables.

Example in code flow

pseudo

event = kafkaConsumer.poll()

tenantConfig = ConfigService.get(event.TenantId, "contact")

features = FeatureBuilder.buildWithConfig(profile, event, tenantConfig)

scores = InferenceClient.scoreWithTenant(event.TenantId, features)

ProfileUpdater.applyTenantRules(updatedProfile, scores, tenantConfig)

NETCELL-EDA Architecture

Platform Design

Thank you