Data Pipeline Service – Detailed User Guide

Document Version: 1.0
Last Updated: February 2026
Service: MyPolicy Data Pipeline (Port 8082)

---

1. Overview

The **Data Pipeline Service** is a consolidated backend service that handles the ingestion, validation, transformation, and processing of insurer policy files. It bridges insurer-provided data (CSV/Excel) with the MyPolicy system by matching policies to existing customers and creating policy records.

1.1 Purpose

- Accept policy file uploads from insurers (Health, Auto, Life)
- Validate file structure against metadata-driven schemas
- Map insurer-specific columns to a canonical schema
- Match each policy row to an existing customer via mobile, email, or PAN
- Create policies in the policy-service and update customer portfolios

1.2 Key Capabilities

Capability	Description
File Upload	CSV and Excel (.xls, .xlsx) up to 50MB
Schema Validation	Metadata-driven validation per insurer type
Field Mapping	Configurable YAML mappings for each insurer
Customer Matching	Mobile → PAN → Email with name/DOB verification

Policy Creation	Creates policies via policy-service API
Verification Failures	Tracks and displays unmatched/failed rows in UI

---

2. Architecture

2.1 Internal Modules

Module	Responsibility
Ingestion	File upload, storage, job creation, status tracking
Metadata	Schema definitions, field mappings (YAML/DB)
Processing	CSV/Excel parsing, field mapping, data massaging
Matching	Customer resolution, policy creation, verification
Portfolio	Consolidated customer+policy view in MongoDB

---

3. Detailed Process Flow

3.1 Upload Phase

1. User selects a policy file (CSV or Excel)
2. User selects insurer type (HEALTH_INSURER, AUTO_INSURER, LIFE_INSURER)
3. File is validated: size (≤50MB), extension (.csv, .xls, .xlsx)
4. Schema validation runs against `insurer-field-mappings.yaml` required columns
5. File is saved to `storage/ingestion/{jobId}.{ext}`

6. Ingestion job created in MongoDB with status `UPLOADED`

3.2 Processing Phase (Triggered by User)

1. Job status transitions to `PROCESSING`
2. File is read and parsed (CSV or Excel)
3. Each row is mapped to canonical schema using `fieldMappings`
4. Data massaging applied (e.g., `normalizeDate`, `normalizeMobile`, `normalizeCurrency`)
5. For each record:
6. **Matching:** Resolve customer via CustomerMatchingRuleService
7. **On success:** Create policy via policy-service, update portfolio
8. **On failure:** Record verification failure (policy number + reason)
9. Job status transitions to `COMPLETED` (or `FAILED` on critical error)

3.3 Matching Logic

Priority order: Mobile → PAN → Email

1. Search customer-service by mobile number
2. If found and name/DOB verify (Levenshtein ≤3 for name): return customerId
3. Else search by PAN
4. Else search by email
5. If multiple identifiers found but name/DOB mismatch: return failure "Verification failed: name or DOB mismatch"
6. If no customer found: return failure "No customer found (mobile/email/PAN)"

---

4. Metadata Configuration

4.1 insurer-field-mappings.yaml

Located at `src/main/resources/metadata/insurer-field-mappings.yaml`

Defines for each insurer: - `requiredSourceColumns` – Columns that must exist in the file - `fieldMappings` – Source column → target field with transforms

Example mapping:

```
HEALTH_INSURER: policyTypes: HEALTH: fieldMappings: - sourceField: "Policy Number" targetField: "policyNumber" dataType: "STRING" - sourceField: "Mobile" targetField: "mobileNumber" transformFunction: "normalizeMobile"
```

4.2 Supported Insurers

Insurer ID	Policy Type	Example File
HEALTH_INSURER	HEALTH	Health_Insurance.csv
AUTO_INSURER	MOTOR	Auto_Insurance.csv
LIFE_INSURER	TERM_LIFE	Life_Insurance.csv

---

5. APIs and Endpoints

5.1 Public API (No Auth)

Method	Endpoint	Description
POST	`/api/public/v1/ingestion/upload`	Upload policy file
GET	`/api/public/v1/ingestion/status/{jobId}`	Get job status
GET	`/api/public/v1/ingestion/jobs`	List all ingestion jobs
POST	`/api/public/v1/ingestion/process/{jobId}`	Trigger processing

5.2 Insurer Portal

- **URL:** `http://localhost:8082/insurer-portal/` or `/`
- **Features:** Upload, process, view status, verification failures

---

6. Data Stores

6.1 MongoDB (ingestion_db)

Collection	Purpose
`ingestion_jobs`	Job metadata, status, total/processed counts, verification failures
`customer_portfolios`	Consolidated customer + policies view

6.2 File Storage

- **Path:** `storage/ingestion/{jobId}.csv` (or .xlsx)
- **Purpose:** Original uploaded file for processing and audit

6.3 PostgreSQL (Optional)

- **Database:** `mypolicy_db`
- **Purpose:** Metadata module (insurer configs, field mappings from DB)
- **Note:** YAML config is primary; DB is optional override

---

7. Prerequisites

7.1 Services Required

Service	Port	Purpose
Customer-service	8081	Customer lookup (mobile, email, PAN)
Policy-service	8085	Policy creation
MongoDB	27017	Ingestion jobs, portfolios

7.2 Customer Data

- Customers must exist in customer-service **before** processing

- Required fields for matching: `mobileNumber` and/or `email` and/or `panNumber`
- Verification fields: `firstName`, `lastName`, `dateOfBirth` (for name/DOB checks)

7.3 Policy File Format

- Must match required columns in `insurer-field-mappings.yaml`
- Values for mobile/email/PAN must match customers in customer-service

---

8. Verification Failures

When matching fails, the pipeline: 1. Records the policy number and failure reason 2. Stores in `IngestionJob.verificationFailures` 3. Displays in the Insurer Portal under "Verification Failures"

Failure reasons: - "No customer found (mobile/email/PAN)" – No customer with matching identifiers - "Verification failed: name or DOB mismatch" – Customer found but name or DOB doesn't match - "Policy creation failed: ..." – Error from policy-service

---

9. Downstream Services (What Comes Next)

9.1 Customer-Service (Upstream Dependency)

Port: 8081
Role: Master data for customers. Data-pipeline calls it to resolve customerId before creating policies.

APIs used: - `GET /api/v1/customers/search/mobile/{mobile}` - `GET /api/v1/customers/search/email/{email}` - `GET /api/v1/customers/search/pan/{pan}`

Database: PostgreSQL `mypolicy_db`, table `customers`

9.2 Policy-Service (Downstream Consumer)

Port: 8085
Role: Stores policies. Data-pipeline creates policies here after successful matching.

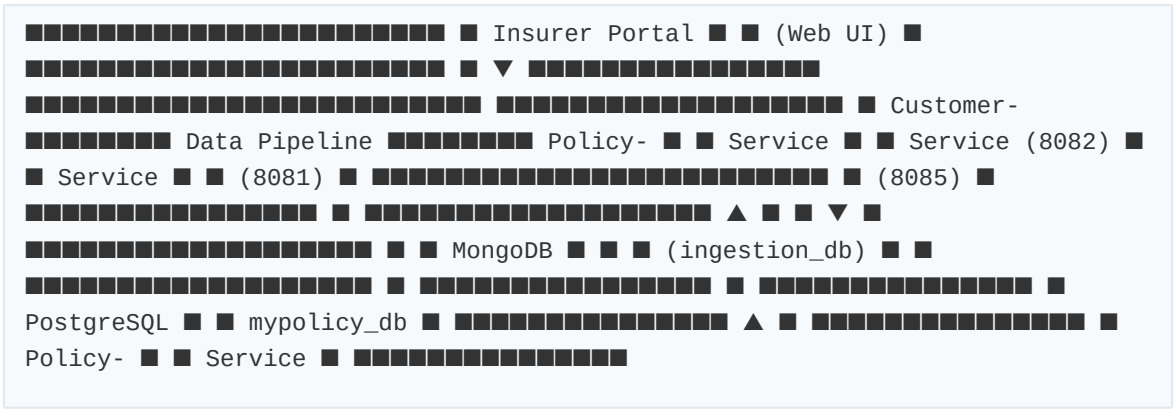
APIs used: - `POST /api/v1/policies` – Create policy (customerId, policyNumber, insurerId, policyType, premiumAmount, sumAssured, etc.)

Database: PostgreSQL `mypolicy_db`, table `policies`

9.3 BFF / Frontend (Optional Consumers)

- **Portfolio API:** `GET /api/public/v1/portfolio` – Returns customer+policy aggregates from MongoDB
- **List Jobs:** `GET /api/public/v1/ingestion/jobs` – For admin/dashboard views

9.4 Service Dependency Diagram



---

10. Startup Order

1. **PostgreSQL** – Create `mypolicy_db` (if using metadata DB)
2. **MongoDB** – Start on localhost:27017
3. **Customer-service** – Port 8081 (add customers first)
4. **Policy-service** – Port 8085
5. **Data Pipeline Service** – Port 8082

With local profile:

```
mvn spring-boot:run "-Dspring-boot.run.profiles=local"
```

- Uses H2 instead of PostgreSQL for metadata
- Still requires MongoDB on localhost:27017

---

11. Key Files Reference

Path	Purpose
`metadata/insurer-field-mappings.yaml`	Insurer schemas and field mappings
`matching/CustomerMatchingRuleService.java`	Matching logic
`matching/MatchingService.java`	Policy creation orchestration
`ingestion/IngestionService.java`	File upload, job lifecycle
`processing/ProcessingService.java`	Parse, map, orchestrate matching
`static/insurer-portal/`	Web UI (index.html, app.js, styles.css)

---

12. Summary

The **Data Pipeline Service** ingests insurer policy files, validates and maps them to a canonical schema, matches each row to a customer via mobile/email/PAN, and creates policies in the policy-service. It depends on **customer-service** for customer data and feeds **policy-service** with newly created policies. Verification failures are tracked and displayed in the Insurer Portal for manual review.

End of Document