Charmi Dedhiya

**Analytics Fox Softwares**

**Business Requirement Document**

**CMS – Data Ingestion Module**

Created / Maintained By: Analytics Fox Software’s

|  |  |
| --- | --- |
| Business Analyst | Charmi Dedhiya |
| Email | cfuria@analyticsfoxsoftware.com |
| Project Lead | Avinash Jha |
| Email | ajha@analyticsfoxsoftwares.com |

Client Information

|  |  |
| --- | --- |
| Name |  |
| Location | Mumbai |
| Contact Name |  |
| Contact Number |  |
| E-Mail |  |

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Remarks | Date | Author | Approved By |
| v1.0 | BRD first draft | 30-07-2025 | Charmi Dedhiya |  |
|  |  |  |  |  |

Contents

[1. Purpose of this Document 4](#_Toc205470836)

[3. Workflow: 6](#_Toc205470837)

[4. Use Cases: 7](#_Toc205470838)

[4.1.1: Data Ingestion Workflow 7](#_Toc205470839)

[4.1.2 Data Source Integration – Manual & SFTP Upload 10](#_Toc205470840)

[4.1.3 Data Validation & Rules Engine: 13](#_Toc205470841)

[4.1.4 Ingestion Logs & Audit Trail: 16](#_Toc205470842)

[4.1.5 Master Data Configuration Alignment 19](#_Toc205470843)

[4.1.6 Error Handling and Notification: 21](#_Toc205470844)

[4.1.7 Upload Source Tracking: 24](#_Toc205470845)

[4.1.8 Retry & Reprocessing Engine: 28](#_Toc205470846)

[4.1.9 Real-time LMS Data Ingestion via API: 31](#_Toc205470847)

[4.1.10 Integration with Monitoring & Observability: 33](#_Toc205470848)

# **1. Purpose of this Document**

**The purpose of the Data Ingestion Workflow is to define a standardized, scalable, and auditable process for acquiring structured data from various internal and external sources (manual uploads, APIs, SFTP, batch systems) into the organization’s data platform. This ingestion pipeline ensures data consistency, quality validation, traceability, and readiness for downstream consumption such as reporting, analytics, and operational automation.**

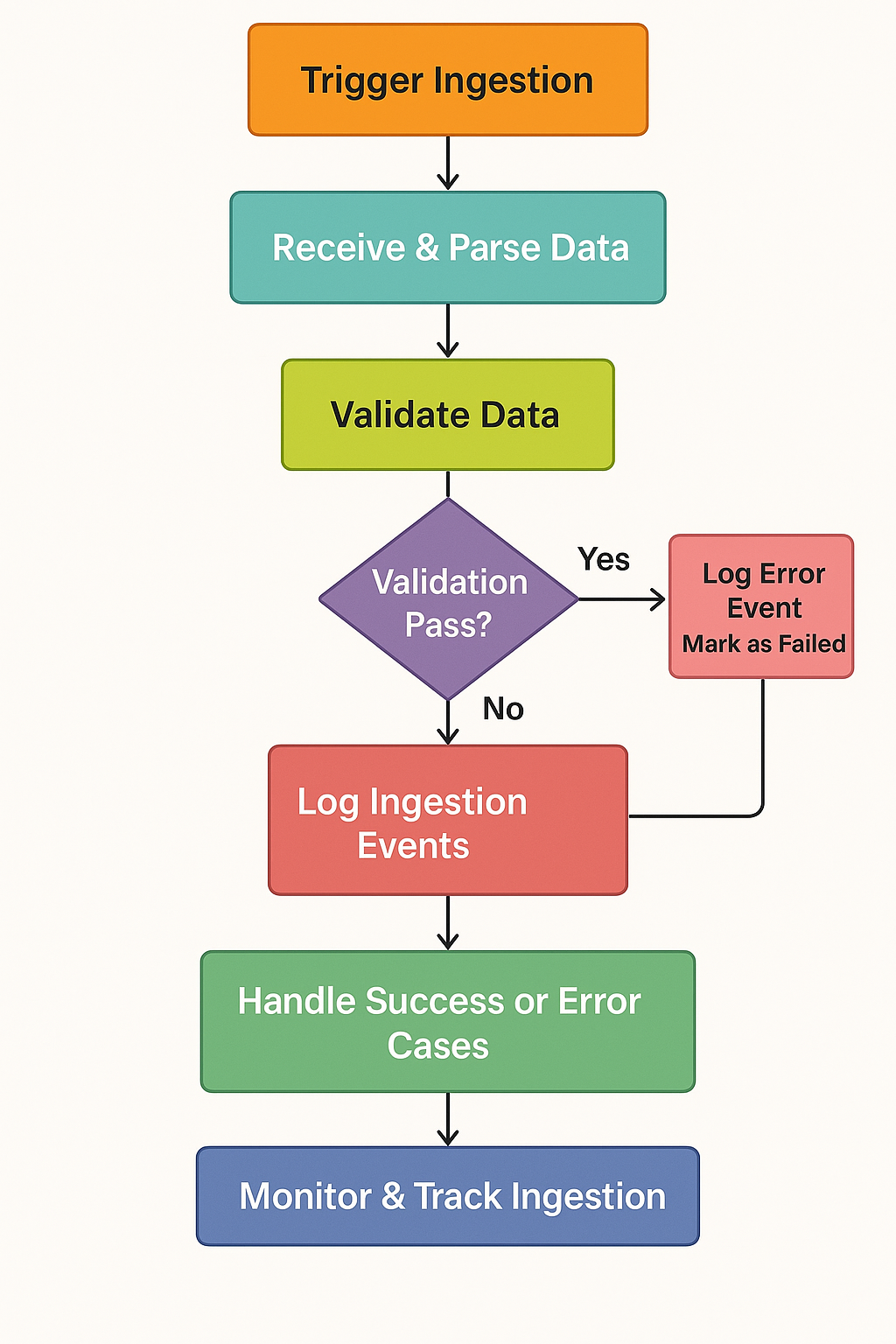
**Objective**

1. Automate and streamline the process of importing data from multiple sources.
2. Enforce data validation rules to ensure data integrity.
3. Maintain a complete audit trail of ingestion activities for compliance and traceability.
4. Provide error handling and real-time notifications on ingestion success or failure.
5. Support both batch-based and real-time data ingestion mechanisms.
6. Ensure seamless integration with monitoring and alerting systems.

**2. Entities:**

|  |  |
| --- | --- |
| **Entity** | **Description** |
| **Ingestion Trigger** | System, scheduler, or user action that initiates the ingestion process. |
| **Source System** | External or internal system that provides the data (e.g., CRM, DWH, API). |
| **Ingestion Engine** | Core module responsible for parsing, validation, and processing of the data. |
| **Parser** | Component that transforms raw input into structured format (JSON, CSV, XML etc). |
| **Validator** | Applies business and data quality rules to ensure correctness of incoming data. |
| **Ingestion Log DB** | Stores metadata and event logs related to ingestion attempts and status. |
| **Error Handler** | Manages validation or system errors; captures and logs them for follow-up. |
| **Notification Service** | Sends alerts/emails to stakeholders on ingestion status (success/failure). |
| **Stakeholders** | Users or systems notified about ingestion outcomes (Ops, Support, Business etc). |
| **Monitoring Dashboard** | Interface to track ingestion performance, errors, retries, and KPIs. |

# **3. Workflow:**



Data Format: [file format](https://analyticsfoxsoftwares-my.sharepoint.com/:x:/p/ajha/EYnRoSD-uyRFtNiS6RxfL8gB4o1THwAYsbKbxMhDfsm6AQ?e=mUBF1U)

# **4. Use Cases:**

### 4.1.1: Data Ingestion Workflow

|  |  |
| --- | --- |
| Use Case ID | UC001 |
| Use Case | Data Ingestion Workflow |
| Actors | * System Admin * Data Provider (Internal/External) * Data Ingestion Engine * Validation Engine * Notification System * Monitoring Layer * API Gateway |
| Description | This use case describes the end-to-end process of ingesting data from multiple sources (manual uploads, API pushes, SFTP batches), validating the data, logging ingestion activities, and handling success/failure cases with stakeholder notification and monitoring integration. |
| Related Use Case ID |  |
| Pre - Condition | * Valid credentials and roles assigned * Source system is pre-integrated * Data format/schema predefined * API keys/SFTP credentials are configured |
| Flow of Events | **High-Level Stages:**   1. **Trigger Ingestion**  * Triggered by manual upload, scheduled SFTP sync, or API call  1. **Receive & Parse Data**  * System parses CSV, JSON, or XML file and maps to schema  1. **Validate Data**  * Field-level validations and rules applied  1. **Log Ingestion Events**  * Event logs created (timestamp, source, user, size, type)  1. **Handle Success/Error Cases**  * If validation passes, mark as successful * If fails, generate error logs with reason codes  1. **Notify Stakeholders**  * Email/SMS/Slack notification for success or error  1. **Monitor & Track Ingestion**  * Logged into monitoring dashboard with status and audit   **Decision Points**   |  |  | | --- | --- | | **Condition** | **Decision Path** | | File/API format valid? | Yes → Proceed to validation No → Reject and notify admin | | All records valid? | Yes → Proceed to staging No → Partial success & error logging | | Ingestion Success? | Yes → Mark completes No → Retry or escalate | | Max Retry Attempts Reached? | Yes → Escalate No → Retry with exponential backoff | |
| Fields and Validation | |  |  |  |  | | --- | --- | --- | --- | | Step No. | Step Name | Description | System/Actor | | 1 | Ingestion Trigger | Ingestion initiated via one of the supported methods: a) Manual File Upload) API Pushc) SFTP Polling | Data Admin / Scheduler | | 2 | Source Identification | Identify source metadata (channel, source name, file/API ID, user) | Ingestion Engine | | 3 | File/API Parsing | Parse file (CSV, Excel) or API payload to extract records. Validate basic format (e.g., delimiter, headers) | Ingestion Engine | | 4 | Validation Engine | Each record is validated against: - Schema (field type, mandatory) - Business rules (e.g., range, match) - Master data lookups | Validation Service | | 5 | Rule Outcome Check | Check if records passed all validations: a) Pass → route to stagingb) Fail → log errors | Validation Service | | 6 | Write to Staging | Validated records written to staging DB / queue / data lake | System DB or Data Layer | | 7 | Write to Error Log | All failed records with detailed error codes & messages logged | Error Logging Service | | 8 | Update Ingestion Status | Job/batch-level metrics (success, failure, source, start/end time) are stored | Ingestion Job Table | | 9 | Notification Dispatch | Summary report sent via Email/Slack: - Status: Success / Partial / Failed - Record counts - Error file link (if any) | Notification Service | | 10 | Dashboard Update | Metrics pushed to monitoring dashboard for observability (Grafana / custom dashboard) | Monitoring Layer | | 11 | Retry Handler (if needed) | If ingestion failed due to system/API issue, job is retried after configurable delay | Retry Queue / Scheduler | |
| Post - Condition | * Valid records stored in target DB * Logs/audit trail generated * Status visible in dashboard |
| Alternative Flow | * In case of schema mismatch, allow user to map columns manually (admin role only) |
| Exceptional Flow | File upload aborted midway due to timeout or corruption → raise error notification and log incomplete status |
| Priority | High |
| Non-Functional Requirement | * System must support 99.9% uptime * Ingestion time should not exceed 1 minute for 10K records * Audit trail must be tamper-proof * Data validation engine must support rules versioning |
| Assumption and Dependency | * Source systems will maintain data format consistency * Notification system (email/SMS) is active and integrated * API Gateway and SFTP server are operational * Monitoring dashboard is available |
| API to be Integrated |  |

### 

### 4.1.2 Data Source Integration – Manual & SFTP Upload

|  |  |
| --- | --- |
| Use Case ID | UC002 |
| Use Case | Data Source Integration – Manual & SFTP Upload |
| Actors | * System Admin * Business/Data Analyst * External Data Provider * Ingestion Engine * Validation Engine * Notification System * Audit Log Service * Monitoring Layer |
| Description | This use case describes how data is ingested into the system via two primary channels – (1) **Manual Upload** of Excel/CSV files through a web interface, and (2) **Scheduled SFTP-based Batch Upload** from external systems. The system will parse, validate, log, and notify stakeholders post ingestion. |
| Related Use Case ID | UC001 |
| Pre - Condition | * File schemas must be predefined and approved * Upload and SFTP credentials are active * Validation rules are pre-configured * UI and SFTP listeners are operational |
| Flow of Events | **Path A: Manual Upload**   1. User logs in and navigates to Upload Data section 2. Uploads Excel/CSV file 3. System parses the file and validates data 4. Success/failure summary is displayed 5. Audit log is updated 6. Stakeholders notified   **Path B: SFTP Batch Upload**   1. External system drops file at SFTP location 2. Scheduler picks up file at scheduled intervals 3. File is parsed and validated automatically 4. System logs ingestion status 5. Error or success message is logged 6. Notifications triggered 7. Monitoring system updates status |
| Fields and Validation | |  |  |  | | --- | --- | --- | | Field | Validation Rule | Mandatory | | File Name | Must follow defined naming convention (e.g. YYYYMMDD\_source.csv) | Yes | | File Format | Must be .csv or .xlsx for manual; .csv for SFTP | Yes | | File Size | ≤ 10MB for Manual, ≤ 20MB for SFTP | Yes | | Schema Match | File headers must match approved schema exactly | Yes | | Encoding | UTF-8 | Yes | | Record Count | Minimum 1, Maximum 10,000 | Yes | |
| Post - Condition | * Data is successfully stored or flagged with detailed error messages * Audit trail updated with timestamps and source * Notification sent to users or admins * Monitoring dashboard reflects ingestion status |
| Alternative Flow | * If column headers do not match, allow admin user to map fields via UI * For SFTP, unmatched schema files are moved to quarantine folder |
| Exceptional Flow | * Upload fails due to timeout or invalid format → system logs the event and alerts admin * SFTP file missing during scheduled check → alert is sent to technical team |
| Priority | High |
| Non-Functional Requirement | * Upload/Sync must complete within 1 minute for 10,000 records * Retry mechanism for failed SFTP sync (up to 3 times) * Secure encrypted transmission (SFTP/HTTPS) * UI should show real-time upload/processing status * Ingestion events must be tracked and queryable |
| Assumption and Dependency | * Schema definitions will be maintained centrally * Admin users are trained to upload and map schema if needed * SFTP credentials and cron scheduler are active * Validation service is running and configured |
| API to be Integrated |  |

### 4.1.3 Data Validation & Rules Engine:

|  |  |
| --- | --- |
| Use Case ID | UC003 |
| Use Case | Data Validation & Rules Engine |
| Actors | * Data Ingestion Engine * System Admin / Business Analyst * Rules Configuration Admin * Validation Engine * Notification Service * Audit Log System |
| Description | This use case enables automated validation of incoming data (via manual upload, SFTP, or API) based on pre-defined rule sets. The engine applies field-level, relational, and conditional validations to ensure data consistency, accuracy, and completeness before ingestion into the database. |
| Related Use Case ID | UC001, UC004 |
| Pre - Condition | * Data file (CSV/Excel/API batch) is received by ingestion engine * Validation rules are configured and active * Schema and field definitions are available |
| Flow of Events | 1. Ingestion engine receives file or data payload 2. Parser extracts field values and metadata 3. Validation Engine loads rule set based on source/scheme 4. Each record is validated for:  * Mandatory fields * Data types * Format compliance (e.g., date, email) * Referential integrity (e.g., master tables) * Custom business rules (e.g., age > 18, amount > 0)  1. Validation results are categorized into:  * Pass * Fail (with error code/message)  1. Success and error logs are written 2. Notification sent to stakeholders (if errors > threshold) 3. Processed file sent to next stage (success queue / error queue) |
| Fields and Validation | |  |  |  |  | | --- | --- | --- | --- | | Field Name | Rule Type | Validation Rule | Mandatory | | CustomerID | Format Check | Alphanumeric, 10 characters max | Yes | | Email | Pattern Match | Must match email regex pattern | Yes | | Date Of Birth | Logical Rule | Must be before current date, age ≥ 18 | Yes | | PAN Number | Custom Regex | Must match PAN format (e.g., [A-Z]{5}\d{4}[A-Z]{1}) | Yes | | Amount | Value Threshold | Must be > 0 and ≤ 10,00,000 | Yes | | State Code | Referential Check | Must match valid codes in Master Table | Yes | |
| Post - Condition | * Data marked as **Validated (Success)** or **Failed** * Errors logged with detailed reasons per row/field * Failed files quarantined or moved to error folder * Pass files sent to ingestion layer |
| Alternative Flow | * Admin configures new validation rules via UI/API * System version-controls each rule set for rollback |
| Exceptional Flow | * Missing rule set for a source file → mark all records as failed, alert admin * System crash during validation → retry once, then fail and log |
| Priority | High |
| Non-Functional Requirement | * Should validate 10,000 records in < 60 seconds * Error rate threshold alerting (e.g., > 5% failure triggers email) * Real-time validation UI feedback for manual uploads * Logs retained for minimum 90 days * Rules must be manageable via Admin interface |
| Assumption and Dependency | * Schema and master data tables are always available * Rule changes are approved before activation * Validation API is scalable and fault-tolerant |
| API to be Integrated |  |

### 4.1.4 Ingestion Logs & Audit Trail:

|  |  |
| --- | --- |
| Use Case ID | UC004 |
| Use Case | Automatic Generation of Case ID |
| Actors | * Ingestion Engine * Validation Engine * System Admin * Monitoring Layer * Compliance/Audit Teams * Notification System * Logging & Audit Microservice |
| Description | This use case captures all data ingestion activities—file uploads, parsing, validation results, rule execution, and outcomes—in detailed logs that are persisted for audit, monitoring, compliance, and troubleshooting. These logs help track when, how, and by whom data was ingested, and whether it was successful or failed, along with detailed metadata. |
| Related Use Case ID | UC003 |
| Pre - Condition | * Logging service is active and available * Ingestion and validation processes are running * File/data source metadata is tagged on entry |
| Flow of Events | * **Ingestion Triggered** – User uploads file or automated job is triggered * **Initial Log Entry** – Ingestion metadata (source, timestamp, file name, size, user ID) is logged * **Validation Log** – Per-record validation status, errors, and rule matches logged * **Status Logging** – Final ingestion result: Success, Partial Fail, or Complete Fail * **Audit Event Logging** – Immutable record created for each transaction (who did what, when) * **Notification Hook** – Trigger notifications if errors exceed thresholds * **Monitoring Update** – Logs pushed to monitoring dashboard/API for visualization * **Log Storage** – Logs persisted in searchable store (e.g., Elastic, SQL) * **Access Control** – Audit logs viewable only by authorized roles |
| Fields and Validation | |  |  | | --- | --- | | Field | Description | | Ingestion ID | Unique identifier for ingestion event | | Timestamp | Date-time of each stage | | Source Type | Manual, SFTP, API | | Source Name | Filename or source system ID | | User ID | Initiator of upload | | File Metadata | Size, format, encoding | | Validation Result | Success, Partial, or Failure | | Errors (if any) | Field-level error details with row number | | Duration | Total time taken from upload to status logging | | Final Status | Completed, Quarantined, Retried, Aborted | | System Logs | Exception logs, retry attempts, parsing issues | | Notification Sent | Timestamp + recipients of alerts | |
| Post - Condition | * Logs and audit events are stored securely * Data is available for monitoring dashboards * Historical ingestion data query able for compliance reports |
| Alternative Flow | * Logs also pushed to external systems via webhook (e.g., Splunk, Kibana, Prometheus) * Separate audit log created for **each sub-component** (upload, validate, notify) |
| Exceptional Flow | * Logging service is down → local backup log stored, retried once service resumes * Log writing fails → ingestion continues but flagged with warning and retry mechanism |
| Priority | High |
| Non-Functional Requirement | * Logs must be searchable by ingestion ID, user, date, status * Immutable storage for audit trail (write-once, read-many) * Must support log retention for 1–3 years * API to export logs in CSV or JSON for audit teams * Secure access based on role and permissions |
| Assumption and Dependency | * Centralized logging infrastructure (e.g., ELK, Splunk) is available * All ingestion subsystems are integrated with logging layer * User authentication is in place to track actions |
| API to be Integrated |  |

### 4.1.5 Master Data Configuration Alignment

|  |  |
| --- | --- |
| Use Case ID | UC005 |
| Use Case | Master Data Configuration Alignment |
| Actors | Admin User, Supervisor |
| Description | To centrally manage and configure all master data required across the Collection Management System (CMS) — including DPD, state, product, communication channel, templates master, product hierarchy, language |
| Related Use Case ID |  |
| Pre - Condition | * Admin is authenticated and authorized to manage master configurations. * Role-based access and menu access for master data modules are enabled. * Base master data structures (e.g., states, languages, product channel). |
| Flow of Events | |  |  |  | | --- | --- | --- | | **Step** | **Actor** | **Action / Description** | | 1 | Admin | Navigates to CMS → Master Configuration | | 2 | Admin | Selects master to configure (e.g., State, DPD, Product) | | 3 | Admin | Clicks “Add New” and fills out required fields | | 4 | System | Validates data: field type, mandatory, uniqueness, dependencies | | 5 | Admin | Submits configuration | | 6 | System | Saves data, marks it as "Active" | | 7 | System | Propagates to all relevant modules consuming the master | |
| Fields and Validation | |  |  |  | | --- | --- | --- | | **Master Type** | **Fields** | **Validation Rules** | | **State Master** | State Code, State Name, State ID, Status | Mandatory, Unique code and ID, Alphabetic name, Status as Active/Inactive | | **DPD Bucket Master** | DPD Bucket ID, Range (e.g., T-6 to T+4), Module (Digital/Call Centre/etc.), Status | Bucket ranges should not overlap, Mandatory fields, Unique bucket ID | | **Channel Master** | Channel ID, Channel Name (SMS, WhatsApp, IVR), Status | Must be among predefined channel types, Unique ID | | **Language Master** | Language Code, Language Name, Script Support, Status | Mandatory, Unique Code, Supported language list (e.g., Marathi, Hindi) | | **Template Master** | Template ID, Channel Type, Language, Message Body, Status | Mandatory fields, Max character limits based on channel, Language match | | **Product Group Master** | Product Group ID, Group Name, Status | Mandatory, Unique ID | | **Product Type Master** | Product Type ID, Group ID (FK), Name, Status | Group ID should exist in Product Group Master | | **Sub-Product Type** | Sub-Type ID, Type ID (FK), Name, Status | Type ID should exist in Product Type Master | | **Product Variant** | Variant ID, Sub-Type ID (FK), Name, Status | Sub-Type must exist | |
| Post - Condition | Updated master reflects in all downstream modules. |
| Alternative Flow | * Admin edits an existing master entry → Fields prefill → Admin updates → Saves. * Admin marks a master entry as "Inactive" → System restricts further usage. |
| Exceptional Flow | * Duplicate value → System shows error. * Invalid references (e.g., non-existent parent product) → System blocks save. * Session timeout or backend error → Save fails with message. |
| Priority |  |
| Non-Functional Requirement | Full audit trail. |
| Assumption and Dependency | * Dependent masters (hierarchies) exist before child masters. * Modules using master should fetch only "Active" values. |
| API to be Integrated | |  | | --- | |  | |

### 4.1.6 Error Handling and Notification:

|  |  |
| --- | --- |
| Use Case ID | UC006 |
| Use Case | Error Handling and Notification Framework for CMS Modules |
| Actors | System, Admin User, Supervisor, Scheduler, Integration API Clients |
| Description | This use case defines the standard way of capturing, logging, notifying, and escalating system and data-level errors encountered during master data updates, batch uploads, or API injections. |
| Related Use Case ID | UC003 |
| Pre - Condition | Module is processing data or external API is invoked. Notification channels (email/SMS/UI alerts) are configured. |
| Flow of Events | 1. System encounters an error (data, system, or API failure) 2. Error is captured in structured error log 3. System classifies error by severity 4. User-facing error message displayed in UI/API response 5. Email/SMS/Push notification triggered based on error type 6. Dashboard shows real-time error logs 7. Optional escalation if unresolved within SLA |
| Fields and Validation | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Field Name | Data Type | Mandatory | Validation Rules | Remarks | | Error ID | UUID / String | Yes | Auto-generated unique identifier | Primary key | | Source | String | Yes | Must match predefined list of CMS modules | e.g., FileUpload, Scheduler | | Error Type | Enum | Yes | [Validation, System, Network, API, Mapping, Authorization] | Categorizes error type | | Error Code | String | Yes | Defined per module | e.g., AGT\_001, DI\_404, AUTH\_403 | | Error Message | Text | Yes | User-readable message | e.g., "Invalid Region Code" | | Root Cause Summary | Text | No | System-diagnosed reason | For internal troubleshooting | | Stack Trace / Debug | Text | No | Only for system or API errors | Captured only in logs, not notifications | | Entity Affected | String | No | e.g., Customer ID / Agent ID / File Name | Entity identifier for error context | | Severity | Enum | Yes | [Info, Warning, Error, Critical] | Determines notification flow | | Retriable | Boolean | Yes | True/False | Used for system-driven retries | | Timestamp | Timestamp | Yes | Auto-populated | When error occurred | | Created By | String | System | Auto-captured user / system name | System actor or user ID | | Resolved | Boolean | No | Default: false | Used for error dashboard tracking | | Resolution Notes | Text | No | Optional remarks from admin | Updated on issue closure |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Scenario | Notification Channel | Recipient | Trigger Timing | Escalation | | File Upload > 20% Errors | Email | Admin, Supervisor | Within 1 min | Yes (if no retry within 2 hours) | | Agent Master Update Failed | UI Alert + Email | Admin | Immediate | No | | API 500 Error | Email + Dashboard Alert | Tech Support | Real-time | Yes | | Invalid Bucket Code | Upload Error Report (CSV) | Admin | Post upload | No | | Network/API Timeout | Log + Retry | System | Retry in 1 min | Retry 3 times before alert | | Notification Service Failure | Log | System | Auto-recovery | Retry queue maintained | |
| Post - Condition | Errors are logged with traceability. Users are notified. Actionable reports and escalations (if required) are triggered. |
| Alternative Flow | Non-blocking warnings are recorded but do not stop batch/API flow. Shown as part of post-processing report. |
| Exceptional Flow | - Logging service unavailable → fallback to local file - Email server down → retry 3 times with exponential backoff |
| Priority | High |
| Non-Functional Requirement | - Logs must be stored for 90 days - Notifications should be sent within 1 minute - Error dashboard must support filtering by module, error code, time |
| Assumption and Dependency | - Email/SMS gateway configured - Logging microservice is online - Notification rules defined per module |
| API to be Integrated | |  | | --- | |  | |

### 4.1.7 Upload Source Tracking:

|  |  |
| --- | --- |
| Use Case ID | UC007 |
| Use Case | Upload Source Tracking for CMS Data Ingestion |
| Actors | Admin, API Client, System Scheduler, Supervisor |
| Description | This use case captures detailed metadata about each data upload—including source type (manual, API, mobile, batch), initiator identity, IP address, device, and timestamp—allowing for end-to-end auditability of all upload activities across modules. |
| Related Use Case ID |  |
| Pre - Condition | Upload attempt initiated via UI, API, mobile, or batch scheduler. Logging module is operational. |
| Flow of Events | 1. Upload initiated by user/system 2. System extracts upload metadata (IP, device, channel, timestamp) 3. Data saved to upload tracking table 4. Linked to transaction log for visibility 5. Available for search in audit/reporting dashboards |
| Fields and Validation | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Field Name | Data Type | Mandatory | Validation Rules | Remarks | | Upload ID | UUID / String | Yes | Auto-generated unique identifier | Primary key | | Source Channel | Enum | Yes | Must be one of [Manual UI, API, Mobile App, Scheduled Job] | Used for analysis and access control | | Uploaded By | String | Yes | Must be a valid User ID or API Client ID | Captured from session/token | | Role at Upload | String | Yes | Derived from user profile | Admin, Supervisor, Agent, etc. | | Upload Timestamp | Timestamp | Yes | System-generated | When upload was triggered | | Source IP Address | String | Yes | Captured automatically | Useful for forensic/audit | | Device Type | String | No | Captured from headers/user-agent | e.g., Windows 10, Android 13 | | Upload File Name | String | No | Captured if upload was file-based | Stored with extension | | Upload Module | Enum | Yes | [Delinquency, Agent Master, Feedback, Payments, etc.] | Defines context | | Upload Status | Enum | Yes | [Success, Partial Success, Failed] | Tied to ingestion outcome | | Total Records | Integer | Yes | Count of records in the batch | For completeness checking | | Valid Records | Integer | Yes | Successfully ingested records | Used in reporting | | Invalid Records | Integer | Yes | Failed validation | Tied to error report log | | Error Log Reference | String (URL) | No | Link to error file/report | Only if upload had failures | | GPS Coordinates | Geo (lat,long) | No | Only captured in Mobile App uploads | Useful for geo-traceability |   example Tracking Scenarios   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Scenario | Source Channel | Uploaded By | Upload Module | Special Notes | | Admin manually uploads Agent Master CSV | Manual UI | admin\_001 | Agent Master | Browser info + IP logged | | Field agent sends visit feedback from app | Mobile App | agent\_548 | Feedback Upload | GPS + mobile device model captured | | CBS pushes delinquency list via API | API | cbs-client-app | Delinquency File | Token used to verify client | | Scheduler runs nightly auto-upload | Scheduled Job | system scheduler | Payment Upload | Cron job ID logged as user proxy | |
| Post - Condition | Upload event and its source metadata are stored and linked to the associated data records or error reports |
| Alternative Flow | - For mobile app: upload tracking includes GPS coordinates - For API: token used to resolve source client identity |
| Exceptional Flow | - Metadata capture fails → upload proceeds, but system logs warning - Source not identifiable → flag as “Unknown” |
| Priority | High |
| Non-Functional Requirement | - Must capture 100% of upload attempts - Audit trail should be immutable - Retain metadata for 90 days minimum |
| Assumption and Dependency | - Logging DB or service is online - Users and API clients are correctly identified |
| API to be Integrated |  |

### 4.1.8 Retry & Reprocessing Engine:

|  |  |
| --- | --- |
| Use Case ID | UC008 |
| Use Case | Retry and Reprocessing Engine |
| Actors | |  | | --- | |  |  |  | | --- | | System Scheduler, Admin User, API Clients, Reprocessing Service | |
| Description | This use case defines the automated and manual mechanisms to retry or reprocess failed records (due to validation, system, or transient issues) in CMS uploads or API ingestion, ensuring end-to-end data completeness. |
| Related Use Case ID |  |
| Pre - Condition | Failed records are logged with identifiable error types; retry/reprocess service is active |
| Flow of Events | |  | | --- | |  |  |  | | --- | | 1. Upload/API request processes records 2. Failed records logged with error type 3. Retry-eligible errors flagged 4. System triggers retry (automated or manual) 5. Retry executed with backoff logic or via admin tool 6. Status updated with success/failure | |
| Fields and Validation | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Field Name | Data Type | Mandatory | Validation Rules | Remarks | | Record ID | UUID / String | Yes | Must exist in error log | Unique to each failed row | | Upload ID | UUID | Yes | Must link to original upload tracking entry | Ensures traceability | | Module Name | String | Yes | Must be one of [Delinquency, Feedback, Payment, Agent Master] | Determines validation context | | Error Code | String | Yes | Must exist in error catalogue | e.g., VAL\_001, SYS\_502 | | Error Type | Enum | Yes | [Validation, System, Network, API, Mapping] | Determines retry eligibility | | Retry Eligible | Boolean | Yes | System-derived; True if retry allowed | Admin can override if needed | | Retry Attempt Count | Integer | Yes | Default = 0; Max = 3 | To prevent infinite loops | | Retry Status | Enum | Yes | [Pending, In Progress, Success, Failed, Skipped] | Status of current or last attempt | | Last Retry Timestamp | Timestamp | No | Captured if retry attempted | For SLA compliance tracking | | Retry Trigger Type | Enum | Yes | [System-Auto, Manual-Admin, API-Client] | Origin of retry command | | Retry Error Log | Text | No | Stores error if retry fails again | Latest failure reason | | Resolution Comment | Text | No | Admin notes if manually resolved | Useful in dashboards | | Created Date | Timestamp | System | Auto-captured | Initial failure time | | Modified Date | Timestamp | System | Auto-updated on retry | Audit-friendly |  |  |  |  |  | | --- | --- | --- | --- | | Error Type | Retryable? | Retry Strategy | Notes | | System Timeout | ✅ | Retry with exponential backoff (e.g., 1m → 2m → 4m) | Auto | | Network Error | ✅ | Retry up to 3 times | API or file-based | | Validation Error | ❌ (default) | Needs manual correction | Admin triggers after fix | | Partial Mapping | ✅ | Retry after master data sync | e.g., Bucket Code added | | API 500 Error | ✅ | Auto-retry in 2 minutes | Monitor for overloads |  |  |  | | --- | --- | | Feature | Description | | Filter by Module / Error / Date | Allows admin to view all retry-eligible failures | | Select & Retry | Retry selected records | | Auto-Fix Suggestions | Shows possible resolutions (e.g., "Add missing Bucket Code") | | Retry Logs | Full history of attempts for each record | | Bulk Retry | Retry all eligible records for a date/module | | Mark as Ignored | Exclude records from further retry |  |  |  | | --- | --- | | Metric | Description | | % of Records Auto-Retried Successfully | Indicates system recovery rate | | Top 5 Non-Retriable Errors | Insights for validation rule improvements | | Retry Load by Module | Capacity planning metric | | Average Time to Final Resolution | SLA metric for data teams | |
| Post - Condition | Records are either successfully processed or remain in “Retry Failed” state with updated error log |
| Alternative Flow | Admin manually selects records from error dashboard and triggers reprocess |
| Exceptional Flow | - Reprocessing engine unavailable → logged for later attempt - Reprocessing causes repeated failure → record marked non-retryable |
| Priority | High |
| Non-Functional Requirement | - Retry must not exceed 3 attempts per record - Reprocessing must not affect already successful records - Retry jobs must be idempotent |
| Assumption and Dependency | - Error log with retry flags must be in place - Reprocessing engine has access to original payload and master data |
| API to be Integrated | |  | | --- | |  | |

### 4.1.9 Real-time LMS Data Ingestion via API:

|  |  |
| --- | --- |
| Use Case ID | UC009 |
| Use Case | Legal Metrics & Dashboards |
| Actors | LMS System, API Gateway, CMS Backend, Admin (for monitoring) |
| Description | The CMS ingests customer and loan data from the LMS through secured API endpoints, enabling dynamic borrower updates, early delinquency flagging, and automated task allocation for collections. |
| Related Use Case ID |  |
| Pre - Condition | LMS system is authenticated via token/API key. API contracts are agreed. Payload must follow schema. |
| Flow of Events | |  | | --- | |  |  |  | | --- | | 1. LMS sends POST request with customer/loan data 2. CMS API authenticates the client 3. Payload is parsed and validated 4. Valid records ingested and stored 5. Invalid records logged with reasons 6. Success/failure response sent to LMS 7. Retry triggered if failure (optional) | |
| Fields and Validation | API Payload Field-Level Validation – Loan Data  Format : [Payload\_Fields\_and\_Validation](https://analyticsfoxsoftwares-my.sharepoint.com/:x:/p/ajha/EYnRoSD-uyRFtNiS6RxfL8gB4o1THwAYsbKbxMhDfsm6AQ?e=nR9cGZ)  Security Considerations   |  |  | | --- | --- | | Area | Specification | | Authentication | OAuth2 token or API Key | | Encryption | HTTPS with TLS 1.2 or above | | Rate Limiting | 1000 RPM per client (configurable) | | Input Sanitization | All fields validated to prevent injection | |
| Post - Condition | Valid loan/customer data from LMS is ingested as per defined format, available in CMS for collections or risk tagging |
| Alternative Flow | LMS calls CMS in batch mode every 1/6/12 hours instead of real-time |
| Exceptional Flow | - Token invalid → HTTP 401 - Schema mismatch → HTTP 400 with error codes - DB down → HTTP 503 |
| Priority | High |
| Non-Functional Requirement | - API should respond within 2 seconds - TLS encryption required - Throughput: 1,000+ records/min supported |
| Assumption and Dependency | - LMS must follow agreed schema - API Gateway, DB, Master Data must be live - Retry engine is enabled for ingestion failures |
| API to be Integrated |  |

### 4.1.10 Integration with Monitoring & Observability:

|  |  |
| --- | --- |
| Use Case ID | UC010 |
| Use Case | Legal Metrics & Dashboards |
| Actors | System (CMS backend), Monitoring Engine (e.g., Prometheus/Grafana, ELK, Dynatrace), Admin/Infra Teams |
| Description | This use case enables CMS to push structured logs, performance metrics, API statuses, and ingestion health indicators to a centralized monitoring layer for real-time observability, alerting, and dashboarding. |
| Related Use Case ID |  |
| Pre - Condition | Monitoring layer is active; endpoints are authenticated. CMS modules are instrumented for event publishing. |
| Flow of Events | |  | | --- | |  |  |  | | --- | | 1. CMS performs action (API call, upload, ingestion, retry, sync) 2. Action triggers log/event/metric 3. Data pushed to monitoring sink (e.g., via webhook or agent) 4. Monitoring system stores, analyzes, and visualizes data 5. Alerts triggered if thresholds are breached 6. Infra/Admins notified | |
| Fields and Validation | Log Event Schema for Monitoring Integration   |  |  |  |  | | --- | --- | --- | --- | | Field Name | Data Type | Mandatory | Description | | Event ID | UUID | Yes | Unique log/event identifier | | Event Type | Enum | Yes | [API\_CALL, FILE\_UPLOAD, RETRY\_ATTEMPT, ERROR, SYNC, LOGIN, AUDIT\_ACTION] | | Source | String | Yes | FileUpload, API, SFTP | | Triggered By | String | Yes | Username or system identifier | | Timestamp | Timestamp | Yes | UTC format | | Status | Enum | Yes | [SUCCESS, WARNING, FAILURE] | | Event Message | Text | Yes | Brief, human-readable summary | | Correlation ID | String | No | For linking logs across modules/API chains | | Hostname/IP | String | No | From where the event originated | | Tags | Map<String,String> | No | Custom key-value pairs (e.g., branch\_code, upload\_id) | |
| Post - Condition | Observability layer receives logs and metrics; dashboards and alerts reflect live system health |
| Alternative Flow | - CMS batches non-critical logs and pushes every X minutes - For mobile clients, limited telemetry is pushed during sync |
| Exceptional Flow | - Monitoring endpoint unreachable → event buffered and retried - Log format invalid → rejected and flagged |
| Priority | High |
| Non-Functional Requirement | - Logging should not impact core CMS performance - Event push latency < 2 sec - 99.9% uptime for metrics pipeline |
| Assumption and Dependency | - Monitoring tool integrated (e.g., ELK, Datadog, Grafana) - Events are schema-compliant and system clocks synced |
| API to be Integrated |  |