**BUSINESS REQUIREMENT SPECIFICATIONS DOCUMENT**

**Data Integration SDK**

| **#** | **Requirement** |
| --- | --- |
| 1. **General Requirements** | |
| A.1 | The Data Integration SDK shall be operated as a standalone or decoupled component from the TMS runtime and shall not be required during real-time transaction processing. The Data Integration SDK instead operates as a configuration and integration utility. |
| A.2 | The Data Integration SDK is intended to be used by any entity implementing Tazama, be it internal system integrators, or any DFSP. |
| A.3 | The Data Integration SDK shall serve as a configuration and integration toolkit to facilitate structured onboarding and ingestion of data into the Tazama platform.   * + Use Case 1: Addition of New Data Elements into Existing Messages.   Enable integration of new fields and data elements into existing Tazama TMS API messages (e.g., enriching Pain.001 with new transaction metadata).   * + Use Case 2: Onboarding of Additional Payment Message Types.   Support the creation, validation, and submission of new message types such as ISO20022 financial messages (Pacs.009, Camt.052), or ISO8583 financial request message (0200), or any equivalent messages via the TMS API using SDK-defined schemas and mapping logic.   * + Use Case 3: Push-Based Data Enrichment via REST APIs.   Support ingestion of external enrichment data pushed to SDK-generated REST API endpoints. This enrichment data will augment the default enrichment database within Tazama.   * + Use Case 4: Pull-Based Data Enrichment via Polling or File-Based Inputs.   Allow the SDK to periodically poll external systems (e.g., via REST APIs or JDBC) or read files (CSV, JSON or convertible to JSON) from defined sources (e.g., file servers, shared drives) to ingest enrichment data into Tazama. |
| 1. **API Endpoint Construction & Deployment for Pushing to DI-SDK** | |
| B.1 | The Data Integration SDK shall provide functionality to construct and deploy new REST API endpoints into the TMS API layer based on a user-provided JSON schema or example message for Use Cases 1 and 2. These endpoints shall be auto-generated based on a user-provided schema or sample message, and shall support ingestion of structured data in both **JSON** and **XML** formats. |
| B.2 | The Data Integration SDK shall provide functionality to construct and deploy new REST API endpoints into the Data Enrichment layer based on a user-provided JSON schema or example message for Use Case 3. |
| B.3 | The Data Integration SDK shall allow the user to define a custom API path (e.g., /sdk/v1/dfsp/transactionLogs) for each endpoint to be created. |
| B.4 | The Data Integration SDK shall support role-based access control by enabling users to associate one or more Keycloak-managed roles to the generated endpoint, restricting access based on privilege configuration. |
| B.5 | The Data Integration SDK shall generate server-side validation logic for the new endpoint based on the JSON schema, including:   * Field presence and data types. * Format constraints (e.g., date, regex, enumerations). * Optional vs required fields. |
| B.6 | The Data Integration SDK shall provide a versioning mechanism or namespace isolation to avoid collisions between user-defined endpoints and system-defined APIs. |
| B.7 | The DISDK shall support the deployment of API endpoints that are designed to meet one of Use Cases 1,2 or 3 into designated deployment zones, such as pre-defined logical environments. These deployment zones shall be configured within the SDK but must exist and be provisioned externally prior to endpoint generation or service deployment. |
| 1. **File-Based Data Enrichment Ingestion for DI-SDK to Pull Data from External Sources** | |
| C.1 | The DISDK shall support pull-based ingestion of enrichment data exclusively from structured CSV files, or files that can be transformed into valid JSON format via a column-mapping utility. |
| C.2 | The DISDK shall support ingestion of structured, text-based files where a column-to-field mapping can be defined to transform the file content into valid JSON.  The SDK shall allow users to specify the delimiter character (e.g., comma, tab, pipe) during configuration, thereby supporting formats such as CSV, TSV, or other delimited text files within a unified ingestion function.  Binary file formats (e.g., XLS, PDF, ZIP) are explicitly out of scope. |
| C.3 | The SDK shall provide a configuration utility that allows users to:   * Define field mappings from CSV columns to JSON fields / JSON composer * Apply static values or transformation rules where required * Preview the resulting JSON structure before ingestion |
| C.4 | The SDK shall support file ingestion only from   * SFTP/FTP sources using either SSH key or password-based auth. * Pull-based public HTTP GET endpoint that returns data in JSON format. (example: https://nfs.punjab.gov.pk/Home/GetJosn?filter= ) |
| C.5 | The SDK will generate code that users can run:   * Fixed interval schedules (e.g., every 30 minutes) using CRON * Manual trigger mode for testing or ad hoc ingestion |
| C.6 | All transformed enrichment data shall be:   * Converted to JSON according to the defined mappings * Validated (if schema is available) * Stored in SDK-designated ArangoDB enrichment collections for later reference by the mapping engine |
| C.7 | If ingestion fails (e.g., malformed CSV, connection issue):   * The file shall not be deleted * The error shall be logged with timestamp and reason |
| C.8 | The DISDK shall support the deployment of Data Pulling Services and Jobs that meet Use Case 4 into designated deployment zones, such as pre-defined logical environments. These deployment zones shall be configured within the SDK but must exist and be provisioned externally prior to endpoint generation or service deployment. |
| 1. **Mapping Utility** | |
| D.1 | The Data Integration SDK shall include a mapping utility that allows users to associate fields from incoming JSON payloads with fields in Tazama’s message model. |
| D.2 | The mapping utility shall support:   * One-to-one field mapping * One-to-many and many-to-one field mappings including transformation of Data such as merging or splitting of fields between source and destination payloads. * Constant value injection (e.g., hardcoded, missing or default fields) |
| D.3 | The utility shall allow users to select from a list of predefined Tazama message templates which are:   * Pain.001 * Pain.013 * Pacs.008 * Pacs.002   Or any other message templates after they have been configured in Tazama using the Data Integration SDK. |
| D.4 | The SDK shall validate the mapped output against the selected Tazama message template before completing the deployment to the TMS API endpoint. |
| D.5 | The SDK shall provide a simulation or dry-run mode where a sample input can be passed through the mapping engine to preview the generated message structure. |
| D.6 | Mappings shall be stored and versioned as SDK configuration artifacts to allow re-use, rollback, and traceability. |
| 1. **Data Destination** | |
| E.1 | The Data Integration SDK shall write structured, ingested, and mapped data to specific collections within the ArangoDB database, which serves as the canonical storage for Tazama’s Operational Data Store (ODS). |
| E.2 | The DISDK shall not write directly to:   * Any external system * TMS runtime storage (outside of the exposed ArangoDB collections)   All data ingestion from the SDK shall terminate within SDK-managed or SDK-created ArangoDB collections, which are accessible by downstream Tazama services. |
| E.3 | The DISDK shall:   * Create collections in ArangoDB where none exist, based on defined schema mappings. * Maintain naming conventions and structural conformity for all new collections. |
| E.4 | All metadata related to:   * Mapping configurations, * Endpoint definitions, * Enrichment sources,   shall be stored in config files within the Data Integration SDK Host Environment. |

Clarification Points:

* Referring to requirement B.1, the transformation capability referred shall enable the ingestion of non-JSON payloads directly into the TMS API via the SDK-generated endpoints, thereby eliminating the need for a separate Payment Platform Adapter (PPA) for the sake of format conversion. To this end, the SDK shall provide a configuration interface or mapping utility to:
  + Define the source structure (e.g., XML tag hierarchy)
  + Map source elements to JSON fields based on the target schema
  + Validate the transformed message against the destination TMS API message structure before submission

All financial messages intended for ingestion by the Data Integration SDK shall be **pushed** into SDK-generated REST API endpoints by upstream systems.

The SDK shall not support:

* Polling of Kafka topics
* Consuming from message queues
* Any custom integration with real-time event streams or proprietary transport protocols

These responsibilities — including integration with specific payment systems or message buses — shall remain out of scope for the SDK and are expected to be handled via a Payment Platform Adapter (PPA) or equivalent component within the implementation-specific layer.