# i-Scanned-Image Table Migration Service

## 1. Overview

The Migration Service is required to move existing data from the i-scanned-image table (on-premises) to AWS. This table contains binary (BLOB) document data and related metadata for the i-Prompt application. Since the application is moving to AWS, we need to migrate this data efficiently and securely while maintaining integrity.

### Key Objectives:

✅ Move BLOB data to Amazon S3  
✅ Move metadata to Amazon DynamoDB  
✅ Ensure secure, high-performance migration  
✅ Maintain data consistency between on-prem and AWS

## 2. Migration Flow & Architecture

### 2.1 High-Level Architecture

On-Premises → Spring Batch → API Connect → Direct Connect → API Gateway → AWS Lambda → S3/DynamoDB

The migration process consists of the following steps:  
1. Extract: Spring Batch reads records from the i-scanned-image table in Oracle DB.  
2. Transform: Data is converted into AWS-compatible format (BLOB → S3 object, metadata → DynamoDB).  
3. Load:  
 - Binary data (BLOBs) → S3  
 - Metadata (GUID, S3 URL, timestamps) → DynamoDB

### 2.2 Component Breakdown

|  |  |
| --- | --- |
| Component | Purpose |
| Spring Batch (on-premises) | Extracts records from Oracle DB and sends them to AWS in batches. |
| API Connect (API-C) [on-prem] | Securely exposes APIs to AWS via Direct Connect. |
| AWS Direct Connect | Provides a high-speed, private network link between on-prem and AWS. |
| API Gateway (AWS) | Handles incoming migration API requests and forwards them to AWS Lambda. |
| AWS Lambda | Processes incoming batch requests, extracts metadata, and uploads files to S3. |
| Amazon S3 | Stores the document binaries (previously BLOBs). |
| Amazon DynamoDB | Stores metadata, including GUIDs, S3 file paths, timestamps, and additional attributes. |

## 3. Migration Implementation

### 3.1 Spring Batch Implementation

• Job Definition: Processes data in chunks (batch size = 50).  
• Parallel Processing: Uses multi-threading (5 concurrent threads) to speed up migration.  
• Error Handling & Retry: Automatically retries failed uploads.  
• Batch REST Calls: Sends batch payloads to API Gateway for efficiency.

### 3.2 API Gateway & Lambda Implementation

• API Gateway Endpoint: Accepts batch migration requests from Spring Batch.  
• Lambda Function: Extracts metadata and uploads binaries to S3.  
• DynamoDB Entry: Stores GUID, metadata fields, and S3 file location.

### 3.3 DynamoDB Table Schema

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| GUID | String (PK) | Unique identifier for the document |
| S3Path | String | Amazon S3 file location |
| CreatedAt | Timestamp | Time of migration |
| DocumentType | String | Type of document |
| SourceSystem | String | Source application (i-Prompt) |

## 4. Data Integrity & Validation

✅ Checksum Verification → Ensures files in S3 match the original BLOBs.  
✅ Transaction Logs → Keeps records of migrated vs. failed records.  
✅ Reconciliation Reports → Validates migrated data between Oracle & AWS.

## 5. Summary & Benefits

• Seamless migration of 6TB+ data from Oracle to AWS.  
• Cost-effective storage with Amazon S3 instead of on-prem BLOBs.  
• Scalable, serverless architecture for future document management.  
• Enhanced search capabilities using metadata in DynamoDB.

## Next Steps

📌 Validate migrated data before switching applications to AWS.  
📌 Monitor S3 & DynamoDB utilization during migration.