BCBSRI-logo

HCLTech | HCL America

HCLTech Confidential

Claims Details API

Design Document

Version # : 1.0

Released on : 10/07/2025

Author : Saikiran Reddy P

## Table of Contents

1. **General Information**

1.1 Executive Summary

1.2 Scope

*1.2.1 In-Scope*

*1.2.2 Out-Scope*

1.3 Document Usage

1.4 Diagrams

*1.4.1 Dataflow Diagram*

*1.4.2 User case Diagram*

1.5 Specification Sheet

2. **Technical Design**

2.1 Input for Claim Details

*2.1.1 Input Json Structure*

*2.1.2 Input request Details*

*2.1.3 Input Request Validation*

*2.1.4 Member API Interaction (New)*

2.2 Output for Claim Details

*2.2.1 Output Json Structure (Internal Transformation)*

*2.2.2 Output Data Transformation (Header/Root Fields)*

*2.2.3 Outputs Data Transformation (Medical line Fields)*

2.3 Audit and Error logging table

3. **Business Rules**

3.1 Claim Filtering Logic (Prime-Pended claims)

*3.1.1 Primary status Filter*

*3.1.2 Mandatory Data check*

*3.1.3 Member data Enrichment*

3.2 Error Handling & Logging

4. **Reference**

4.1 PSCE\_PENDED\_CLAIMS\_OUTBOUND (Snowflake Table DDL - Target Data)

4.2 PSCE\_AUDIT\_ERROR\_LOG (Snowflake Table DDL - Transaction Log and Retry)

5. **Amendment History**

Software Design Document (SDD) – PCSE: Outbound Claims Processing

# 1. General Information

## 1.1 Executive Summary

This document defines the technical architecture and design specifications for the Prime Claims Extract (PSCE) Outbound pipeline. The primary objective of this pipeline is to securely process high-volume claims data originating from the Facets system; this pipeline uses QueueProcessBlobFile (Data Mover) to retrieve full claims file using a provided URL/Path and copy it to a new Azure storage Container. BlobProcessSnowflakeData is triggered by new file, filter records that are in a 'Pended for PRIME Review' status, enrich the data by fetching corresponding member details from a Member API, and load the required data elements into a Snowflake data warehouse table for downstream processing by Prime Therapeutics. The solution is implemented using two decoupled, event-driven Azure Functions for high scalability and fault tolerance.

## 1.2 *Scope*

The PSCE Outbound pipeline focuses on moving, validating, filtering, enriching, and persisting claim data pointers and filtered results.

*1.2.1 In-Scope*:

● QueueProcessBolbFile(Queue to Blob Function): Reading raw message data from the Facets Outbound Queue, retrieving the full claims file from the source storage container using the path and copying the full claims file from the source storage container using the path and copying the full claims file into a designated Azure Blob Storage container.

● BlobProcessSnowflakeData (Blob to Snowflake Function): Triggered by the new full claims file in the destination blob, reading and validating the payload structure, applying the 'Pended' filtering business rule, calling an external Member API for data enrichment, merging the data, and performing bulk insertion into the Snowflake target table.

● Logging & Auditing: Implementation of detailed audit and error logging using Azure Application Insights (the designated Azure Error Log Service). Only ERROR logs will persist with the centralized Snowflake Audit Log Table for operational review and retry management. Successful audit logs will remain in Application Insights.

● Target Database Schema: Insertion of data into the DOPDEV.MPS\_PRM.PSCE\_PENDED\_CLAIMS\_OUTBOUND Snowflake table.

*1.2.2* *Out-of-Scope*

● The generation or creation of the initial message and data URL by the Facets system.

● The internal claims processing, review, and edits performed by Prime Therapeutics after data is loaded into Snowflake.

● The final SFTP transfer of data from Snowflake to Prime.

● Facets-side configuration or adjudication logic that determines the initial 'Pended' status.

## 1.3 *Document Usage*

* **Developers:** To guide the implementation of Azure Functions, authentication and data transformation logic.
* **Quality Assurance (QA):** To define test cases, focusing on input validation, business rule adherence (filtering logic) and Member API integration.
* **Support/Operations**: To understand the end-to-end process, dependencies and to interpret audit and error logs in application Insights, and to use the Audit Log Table for monitoring and manual/automated retry of failed records.

## 1.4 Diagrams

### *1.4.1* *Data Flow Diagram*

Description: The process begins with a message (containing a data URL/Path) placed in the Azure Queue (Source Queue) by the Facets system. QueueProcessBlobFile consumes this message, fetches the large claims Json file from the source. And writes the full file to the azure Blob storage Destination container (F2 trigger). This triggers BlobProcessSnowflakeData which then reads the full claim Json payload from the destination container. BlobProcessSnowflakeData filters the claims, calls an external member API for each pended claim, merges the member data and loads the qualifying records into snowflake. Application insights capture all audit and error logs from both functions and only Error logs will be persisted to the centralized Snowflake Audit Log table for operational review and retry management. Successful audit logs will remain in application insights.

Reference: Please refer to the companion file pcse\_outbound\_flowdiagram.html for a detailed, visual representation for this technical architecture, including the steps for QueueProcessBolbFile, F2 and member API enrichment.

A diagram of a software flow

AI-generated content may be incorrect.

### *1.4.2* *Use Case Diagram*

## Description:

## ● Actors: Facets System, Azure Functions Pipeline, Member API, Snowflake DB, Operations Team (monitoring).

## ● Use Cases: Receive Claim Pointer, Retrieve Full claim Data (Moved to QueueProcessBolbFile). Store Full claim Data (New container), Trigger Claim processing, Filter Pended claims, Enrich Claim with member data (New), Insert into Snowflake, Log Operational events (Audit/Error), Track and Manage Failed Transactions.

## 1.5 *Specification Sheet*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Type** | **Trigger** | **Purpose** | **Output/Target** |
| QueueProcessBlobFile | Azure Function (Queue Trigger) | Azure Storage Queue (facets-outbound-queue) | Fetch full claim file from source store and copy it into a new container. | Azure Blob Storage (Destination Container) =FULL FILE, Snowflake Audit Log (error only). |
| BlobProcessSnowflakeData | Azure Function (Blob Trigger) | Azure Blob Storage (Destination Container)-Full File | Read full claim data, Filter, Call Member API for enrichment, Transform, Load. | Snowflake DB (PSCE\_PENDED\_ CLAIMS\_OUTBO UND), Snowflake Audit Log (on error only) |
| Logging | Azure Service | Synchronous/Asynchronous calls from QueueProcessBolbFile & F2 | Capture Audit/Error Logs. | Azure Application Insights, Snowflake Audit Log (Critical Errors Only) |

# 2.Technical Design

#### 2.1 Input for Claim Details

#### The process is initiated by the Facets system placing a message into the Source Queue. This message is not the full claims data, but a pointer to the location of the large data file.

#### 2.1.1 Input JSON Structure

#### A. Queue Message (Pointer/Metadata - Input to QueueProcessBolbFile): This is the raw message consumed by QueueProcessBlobFile. QueueProcessBlobFile uses dataUrl to fetch the full data payload.

#### {

#### "messageId": "FACETS-MSG-20251007-001234",

#### "timestamp": "2025-10-07T14:30:00Z",

#### "dataUrl":

#### "[https://claimsdatastore.azurewebsites.net/api/claimsextract/20251007 /001234.json] (https://claimsdatastore.azurewebsites.net/api/claimsextr act/20251007/001234.json)”,

#### "sourceSystem": "Facets"

#### }

#### B. Large Claims JSON Payload (Output from QueueProcessBolbFile / Input to F2):

#### This is the full claims payload written to the destination blob by QueueProcessBlobFile and read directly by BlobProcessSnowflakeData.

#### {

#### "extractId": "001234",

#### "claimsBatch": [

#### {

#### "claimId": "C001001A",

#### "claimLineNumber": 1,

#### "facetsStatus": "Pend for PRIME Review",

#### "memberId": "M00987",

#### "NDC": "00000000101",

#### "quantityDispensed": 50.0,

#### "unitOfMeasure": "ML",

#### "primaryDiagnosis": "J45.909",

#### "procedureCode": "J9001",

#### "prescribingNPI": "1234567890",

#### "renderingNPI": "0987654321",

#### "serviceDate": "2025-09-01",

#### "totalCharges": 1500.50,

#### "groupId": "G100"

#### },

#### // ... many more claim records, including non-pended statuses

#### ]

#### }

#### 2.1.2 Input Request Details

#### QueueProcessBlobFile:

#### Triggered directly by the message arriving in the Azure Queue.

#### Retrieve the full claim payload from the dataUrl using an authenticated HTTP GET request.

#### Writes the full payload to the destination Blob storage container.

#### BlobProcessSnowflakeData:

#### Triggered by the full claims file write operation from QueueProcessBlobFile.

#### Reads the local full file directly from the Blob storage container (F2 trigger).

#### 2.1.3 Input - Request Validation

#### QueueProcessBlobFile Validation (Queue Message):

#### Check for existence and non-empty values for messageId and dataUrl.

#### Verify successful retrieval of the full data payload (HTTP status 200 ok).

#### Verify Successful write operation to the destination blob.

#### BlobProcessSnowflakeData Validation (Claims JSON Payload):

#### Ensure the fetched JSON is valid (JSON parsing successful).

#### Verify that the root element contains the claims Batch array.

#### For each claim in claimsBatch, validate the presence of the mandatory fields required by the Snowflake schema (e.g., claimId, claimLineNumber, facetsStatus).

#### 2.1.4 Member API Interaction (New)

#### BlobProcessSnowflakeData will execute an authenticated HTTP GET request to the Member API for every claim line that passes the Pended status filter (Rule 3.1.1).

#### A. Member API Request (Input to API):

#### ● Endpoint: [MEMBER\_API\_URL]/api/members/memberId}

#### ● Parameter: memberId (sourced from claimsBatch[i]. memberId)

#### ● Authentication: Must use an Azure Managed Identity or Service Principal for secure access.

B. Member API Response (Output from API):

* + - * HTTP Status: Expected 200 OK for success.
      * Payload Example:

{

"memberId": "M00987",

"firstName": "Jane",

"lastName": "Doe",

"dateOfBirth": "1990-01-15",

"memberStatus": "Active"

}

**2.2 Output for Claim Details**

The primary output of the entire process is the structured data record inserted into the Snowflake table, now inclusive of Member API data.

*2.2.1 Output JSON Structure (Internal Transformation)*

After filtering, enrichment, and transformation, the internal record structure (prior to insertion) will strictly adhere to the Snowflake table schema definition.

*2.2.2 Output Data Transformation (Header/Root Fields):*

|  |  |  |  |
| --- | --- | --- | --- |
| Source Field (Claims JSON) | Target Snowflake Column | Data Type | Transformation Logic |
| claimsBatch[i]. claimId | CLAIM\_ID | VARCHAR (20) | Direct mapping. |
| claimsBatch[i]. claimLineNumber | CLAIM\_LINE\_NUMBER | INTEGER | Direct mapping. |
| claimsBatch[i]. memberId | MEMBER\_ID | VARCHAR (12) | Direct mapping. |
| firstName | PATIENT\_FIRST\_NAME | VARCHAR (15) | Direct mapping. |
| lastName | PATIENT\_MIDDLE\_NAME | VARCHAR (15) | Direct mapping. |
| dateOfBirth | PATIENT\_LAST\_NAME | VARCHAR (25) | Direct mapping and date formatting. |
|  | PATIENT\_DATE\_OF\_BIRTH | DATE | Direct mapping. |
|  | PRIMARY\_COVERAGE\_INDICATOR | VARCHAR (1) |  |
|  | PROVIDER\_TAX\_ID\_NUMBER | VARCHAR (10) |  |
|  | RENDERING\_PROVIDER\_NAME | VARCHAR (36) |  |
|  | RENDERING\_PROVIDER\_ADDRESS\_1 | VARCHAR (36) |  |
|  | RENDERING\_PROVIDER\_ADDRESS\_2 | VARCHAR (36) |  |
|  | RENDERING\_PROVIDER\_CITY | VARCHAR (24) |  |
|  | RENDERING\_PROVIDER\_STATE | VARCHAR (2) |  |
|  | RENDERING\_PROVIDER\_ZIP | VARCHAR (10) |  |
|  | RENDERING\_PROVIDER\_NPI | VARCHAR (12) |  |
|  | PAY\_TO\_PROVIDER\_NAME | VARCHAR (36) |  |
|  | PAY\_TO\_PROVIDER\_ADDRESS\_1 | VARCHAR (36) |  |
|  | PAY\_TO\_PROVIDER\_ADDRESS\_2 | VARCHAR (36) |  |
|  | PAY\_TO\_PROVIDER\_CITY | VARCHAR (24) |  |
|  | PAY\_TO\_PROVIDER\_STATE | VARCHAR (2) |  |
|  | PAY\_TO\_PROVIDER\_ZIP\_CODE | VARCHAR (10) |  |
|  | PAY\_TO\_PROVIDER\_PRIMARY\_NPI | VARCHAR (12) |  |
|  | CLAIM\_RECEIVED\_DATE | DATE |  |
|  | SERVICE\_DATE\_FROM DATE | DATE |  |
|  | SERVICE\_DATE\_TO DATE | DATE |  |
|  | PLACE\_OF\_SERVICE | VARCHAR (2) |  |
|  | TYPE\_OF\_BILL | VARCHAR (4) |  |
|  | DIAGNOSIS\_CODE\_1 | VARCHAR (10) |  |
|  | DIAGNOSIS\_CODE\_2 | VARCHAR (10) |  |
|  | DIAGNOSIS\_CODE\_3 | VARCHAR (10) |  |
|  | DIAGNOSIS\_CODE\_4 | VARCHAR (10) |  |
|  | PROCEDURE\_CODE | VARCHAR (8) |  |
|  | MODIFIER\_1 | VARCHAR (2) |  |
|  | MODIFIER\_2 | VARCHAR (2) |  |
|  | MODIFIER\_3 | VARCHAR (2) |  |
|  | MODIFIER\_4 | VARCHAR (2) |  |
|  | HCPCS\_UNITS\_QUANTITY | NUMERIC (9, 0) |  |
|  | NDC\_CODE | VARCHAR (11) |  |
|  | NDC\_QUANTITY | NUMERIC (15, 2) |  |
|  | NDC\_UOM | VARCHAR (2) |  |
|  | BILLED\_AMOUNT | NUMERIC (9, 0) |  |
|  | ALLOWED\_AMOUNT | NUMERIC (9, 0) |  |
|  | CLIENT\_CLAIM\_LINE\_STATUS | VARCHAR (1) |  |
|  | CLIENT\_DENIAL\_CODE | VARCHAR (5) |  |
|  | ADJUSTMENT\_INDICATOR | VARCHAR (1) |  |
|  | ORIGINAL\_CLAIM\_NUMBER | VARCHAR (20) |  |
|  | ITS\_INDICATOR | VARCHAR (1) |  |
|  | SCCF\_NUMBER | VARCHAR (17) |  |
|  | RENDERING\_PROVIDER\_PAR\_STATUS | VARCHAR (1) |  |
|  | USER\_DEFINED\_4 | VARCHAR (20) |  |
|  | PSCE\_EXTRACT\_DATETIME | TIMESTAMP\_NTZ (9) |  |
|  | AZURE\_BLOB\_SOURCE\_URL | VARCHAR (500) |  |
|  | AZURE\_FUNCTION\_RUN\_ID | VARCHAR (100) |  |
|  | CREATED\_DATETIME | TIMESTAMP\_NTZ (9) |  |
|  | PROCESSED\_DATETIME | TIMESTAMP\_NTZ (9) |  |

*2.2.3 Output Data Transformation (Medical Line Fields)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Source Field (Claims JSON)** | **Target Snowflake Column** | **Data Type** | **Transformation Logic** |
| claimsBatch[i]. NDC | NDC\_CODE | VARCHAR (11) | Direct mapping. |
| claimsBatch[i]. quantity Dispensed | QUANTITY\_DISPENSED | NUMERIC (18, 2) | Direct mapping. |
| claimsBatch[i].unitofMeasure | QUANTITY\_UNIT\_OF\_ MEASURE | VARCHAR (10) | Direct mapping. |
| claimsBatch[i].primaryDiagnosis | DIAGNOSIS\_CODE\_PRIMARY | VARCHAR (10) | Direct mapping (ICD-10). |
| claimsBatch[i].procedureCode | PROCEDURE\_CODE | VARCHAR (20) | Direct mapping (CPT/HCPCS). |
| claimsBatch[i].prescribingNPI | PRESCRIBING\_PROVIDER\_NPI | VARCHAR (20) | Direct mapping. |
| claimsBatch[i].renderingNPI | RENDERING\_PROVIDER\_NPI | VARCHAR (20) | Direct mapping. |
| claimsBatch[i]. facetsStatus | FACETS\_PEND\_STATUS | VARCHAR (50) | Direct mapping. |
| claimsBatch[i].serviceDate | SERVICE\_DATE | DATE | Direct mapping and date formatting. |
| claimsBatch[i].totalCharges | TOTAL\_CHARGES | NUMERIC(18, 2) | Direct mapping. |
| Calculated | PSCE\_EXTRACT\_DATETIME | TIMESTAMP\_NTZ(9) | F2 Function execution timestamp. |

**2.3 Audit and Error Logging Table:**

To enable comprehensive error tracking and a retry mechanism for failed claims, the DOPDEV.MPS\_PRM.PSCE\_AUDIT\_ERROR\_LOG table will exclusively capture records where a non-recoverable error occurred (e.g., F1002, F2004) or a key failure milestone was reached. Successful transactions (AUDIT logs) are primarily recorded in Application Insights.

**Logging Context Determination:**

The DATA\_URL, CLAIM\_ID, and CLAIM\_LINE\_NUMBER fields are critical for determining the retry scope. Their population rules are as follows:

(Batch/Process Errors - F100X): Errors occurring here affect the entire batch (e.g., failed to read queue message, failed to write blob).

○ DATA\_URL will be populated (if available from the queue message).

CLAIM\_ID and CLAIM\_LINE\_NUMBER will be set to NULL. The retry mechanism must re-initiate the process for the entire batch URL.

2. **BlobProcessSnowflakeData** (Claim/Line Errors - F2004, F2005): Errors occurring here affect specific claim records after the payload has been fetched and parsed.

○ DATA\_URL will be populated.

○ CLAIM\_ID and CLAIM\_LINE\_NUMBER will be populated with the details of the failing record. The retry mechanism can target this specific claim line.

**Retry Flow**:

1. Identification: Operations/Support team queries the PSCE\_AUDIT\_ERROR\_LOG table for records where LOG\_TYPE = 'ERROR', IS\_RESOLVED = FALSE, and RETRY\_COUNT < 3 (configurable limit).

2. Correction: The root cause (e.g., a data issue, a temporary network failure) is investigated and corrected externally.

3. Trigger: An automated process or a manual tool reads the DATA\_URL and CLAIM\_ID from the failed log record and retriggers the BlobProcessSnowflakeData logic for that specific claim or batch.

4. Update Log: Upon successful re-processing, the IS\_RESOLVED flag is set to TRUE and RETRY\_COUNT is incremented.

**3 BUSINESS RULES**

**3.1 Claim Filtering Logic (Prime-Pended Claims)**

BlobProcessSnowflakeData is responsible for applying the core business filter to the fetched claims payload.

*Rule 3.1.1 (Primary Status Filter):*

* Condition: Only claim lines where the source field claimsBatch[i].facetsStatus is an exact match for the configured Pended Status.
* Target Value (as per SRS): 'Pend for PRIME Review'
* Action: If status matches, the record is flagged for transformation and insertion into Snowflake. If not, the record is skipped, and a low-level audit log entry is made (CLAIM\_SKIPPED, reason: Status Not Pended) only in Application Insights. No entry is written to the Snowflake Audit Log Table. Rule

*3.1.2 (Mandatory Data Check):*

* Condition: All Pended claims must contain non-null and non-empty values for: NDC\_CODE, QUANTITY\_DISPENSED, DIAGNOSIS\_CODE\_PRIMARY, and PROCEDURE\_CODE.
* Action: If a Pended claim fails this check, it is not inserted into Snowflake. An ERROR log entry is created (F2004: MISSING\_MANDATORY\_DATA), detailing the missing fields and the CLAIM\_ID. This ERROR log is written to both Application Insights and the Audit Log Table, with IS\_RESOLVED set to FALSE. Rule

*3.1.3 (Member Data Enrichment):*

* Condition: For every claim line successfully filtered by Rule 3.1.1, BlobProcessSnowflakeData must successfully retrieve member details from the Member API.
* Action: If the Member API returns an HTTP status code other than 200, or if the returned payload is invalid/missing critical fields, the claim line is not inserted into Snowflake. An ERROR log entry is created (F2006: MEMBER\_API\_FETCH\_FAILURE), detailing the API call error. This ERROR log is written to both Application Insights and the Audit Log Table, with IS\_RESOLVED set to FALSE.

**3.2 Error Handling & Logging**

All logging will utilize Azure Application Insights for structured, centralized error and audit tracking, satisfying the 'Azure Error Log Service' requirement. Critical logs (all errors) will be persisted to the PSCE\_AUDIT\_ERROR\_LOG Snowflake table. Successful batch-level and record-level AUDIT logs will reside solely in Application Insights.

1. Audit Logging (Type: AUDIT):

Used for successful operational steps to track progress and performance. These logs are written ONLY to Azure Application Insights. | Field | Value | Example Event | | :--- | :--- | :--- | | LogType | AUDIT | Function start, Blob write success, X claims filtered, Snowflake insert success. | | ProcessStep | BLOB\_WRITE\_SUCCESS, CLAIM\_FILTERED, SNOWFLAKE\_INSERT | | | CorrelationId | FACETS-MSG-20251007-001234 | The queue/blob message ID. | | RecordsProcessed | 4500 | Count of claims processed/filtered/inserted.

1. Error Logging (Type: ERROR):

Used for process failures requiring investigation and potential manual intervention. Each error entry must include a custom ErrorCode. Errors are written to both Application Insights and persist to the Audit Log Table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Error Code** | **Function** | **Description** | **Retry Context Fields Populated** |
| F1001 | F1 | Queue Message Read Failure. | DATA\_URL (if available); CLAIM\_ID=NULL; CLAIM\_LINE\_NUMBER=NULL |
| F1002 | F1 | Blob Write/Connection Failure. | DATA\_URL (from input); CLAIM\_ID=NULL; CLAIM\_LINE\_NUMBER=NULL |
| F2001 | F2 | Input JSON (Metadata) Validation Error. | DATA\_URL; CLAIM\_ID=NULL; CLAIM\_LINE\_NUMBER=NULL |
| F2002 | F2 | Data URL Fetch Failure (HTTP Error). | DATA\_URL; CLAIM\_ID=NULL; CLAIM\_LINE\_NUMBER=NULL |
| F2003 | F2 | Claims JSON Payload Structure Invalid (Parsing error). | DATA\_URL; CLAIM\_ID=NULL; CLAIM\_LINE\_NUMBER=NULL |
| F2004 | F2 | Missing Mandatory Data (Rule 3.1.2). | DATA\_URL, CLAIM\_ID, CLAIM\_LINE\_NUMBER |
| F2005 | F2 | Snowflake Insertion/SQL Error. | DATA\_URL,CLAIM\_ID, CLAIM\_LINE\_NUMBER (of the specific record that failed) |
| F2006 | F2 | Member API Fetch Failure (Rule 3.1.3). | DATA\_URL,CLAIM\_ID, CLAIM\_LINE\_NUMBER |

**4. REFERENCE**

**4.1 PSCE\_PENDED\_CLAIMS\_OUTBOUND (Snowflake Table DDL - Target Data)**

CREATE TABLE DOPDEV.MPS\_PRM.PSCE\_PENDED\_CLAIMS\_OUTBOUND (

-- Unique Key & Identifiers

CLAIM\_ID VARCHAR(20) NOT NULL, -- Sender Claim Number

CLAIM\_LINE\_NUMBER INTEGER NOT NULL, -- Claim Line Number

MEMBER\_ID VARCHAR(12) NOT NULL,

-- Patient Information

PATIENT\_FIRST\_NAME VARCHAR(15) NOT NULL,

PATIENT\_MIDDLE\_NAME VARCHAR(15),

PATIENT\_LAST\_NAME VARCHAR(25) NOT NULL,

PATIENT\_DATE\_OF\_BIRTH DATE NOT NULL,

-- Coverage & Provider Details

PRIMARY\_COVERAGE\_INDICATOR VARCHAR(1),

PROVIDER\_TAX\_ID\_NUMBER VARCHAR(10) NOT NULL,

RENDERING\_PROVIDER\_NAME VARCHAR(36) NOT NULL,

RENDERING\_PROVIDER\_ADDRESS\_1 VARCHAR(36) NOT NULL,

RENDERING\_PROVIDER\_ADDRESS\_2 VARCHAR(36),

RENDERING\_PROVIDER\_CITY VARCHAR(24) NOT NULL,

RENDERING\_PROVIDER\_STATE VARCHAR(2) NOT NULL,

RENDERING\_PROVIDER\_ZIP VARCHAR(10) NOT NULL,

RENDERING\_PROVIDER\_NPI VARCHAR(12),

PAY\_TO\_PROVIDER\_NAME VARCHAR(36) NOT NULL,

PAY\_TO\_PROVIDER\_ADDRESS\_1 VARCHAR(36) NOT NULL,

PAY\_TO\_PROVIDER\_ADDRESS\_2 VARCHAR(36),

PAY\_TO\_PROVIDER\_CITY VARCHAR(24) NOT NULL,

PAY\_TO\_PROVIDER\_STATE VARCHAR(2) NOT NULL,

PAY\_TO\_PROVIDER\_ZIP\_CODE VARCHAR(10) NOT NULL,

PAY\_TO\_PROVIDER\_PRIMARY\_NPI VARCHAR(12),

-- Claim Dates

CLAIM\_RECEIVED\_DATE DATE NOT NULL,

SERVICE\_DATE\_FROM DATE NOT NULL,

SERVICE\_DATE\_TO DATE NOT NULL,

-- Claim Details

PLACE\_OF\_SERVICE VARCHAR(2) NOT NULL,

TYPE\_OF\_BILL VARCHAR(4),

DIAGNOSIS\_CODE\_1 VARCHAR(10) NOT NULL,

DIAGNOSIS\_CODE\_2 VARCHAR(10),

DIAGNOSIS\_CODE\_3 VARCHAR(10),

DIAGNOSIS\_CODE\_4 VARCHAR(10),

PROCEDURE\_CODE VARCHAR(8) NOT NULL,

MODIFIER\_1 VARCHAR(2),

MODIFIER\_2 VARCHAR(2),

MODIFIER\_3 VARCHAR(2),

MODIFIER\_4 VARCHAR(2),

HCPCS\_UNITS\_QUANTITY NUMERIC(9, 0) NOT NULL,

NDC\_CODE VARCHAR(11),

NDC\_QUANTITY NUMERIC(15, 2),

NDC\_UOM VARCHAR(2),

BILLED\_AMOUNT NUMERIC(9, 0) NOT NULL,

ALLOWED\_AMOUNT NUMERIC(9, 0),

CLIENT\_CLAIM\_LINE\_STATUS VARCHAR(1) NOT NULL,

CLIENT\_DENIAL\_CODE VARCHAR(5),

ADJUSTMENT\_INDICATOR VARCHAR(1) NOT NULL,

ORIGINAL\_CLAIM\_NUMBER VARCHAR(20),

ITS\_INDICATOR VARCHAR(1) NOT NULL,

SCCF\_NUMBER VARCHAR(17) NOT NULL,

RENDERING\_PROVIDER\_PAR\_STATUS VARCHAR(1) NOT NULL,

USER\_DEFINED\_4 VARCHAR(20),

-- Audit and Processing Metadata

PSCE\_EXTRACT\_DATETIME TIMESTAMP\_NTZ(9) NOT NULL,

AZURE\_BLOB\_SOURCE\_URL VARCHAR(500) NOT NULL,

AZURE\_FUNCTION\_RUN\_ID VARCHAR(100) NOT NULL,

CREATED\_DATETIME TIMESTAMP\_NTZ(9) NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

PROCESSED\_DATETIME TIMESTAMP\_NTZ(9)

-- Constraints

-- PRIMARY KEY (CLAIM\_ID, CLAIM\_LINE\_NUMBER)

);

**4.2 PSCE\_AUDIT\_ERROR\_LOG (Snowflake Table DDL - Transaction Log and Retry)**

This table captures all critical events for monitoring and provides the necessary context for re-triggering failed transactions.

CREATE TABLE DOPDEV.MPS\_PRM.PSCE\_AUDIT\_ERROR\_LOG(

-- Primary Key and Timing

LOG\_ID VARCHAR(36) NOT NULL PRIMARY

KEY, -- UUID/GUID for PK

LOG\_DATETIME TIMESTAMP\_NTZ(9) DEFAULT

CURRENT\_TIMESTAMP(),

-- Process Identification

LOG\_TYPE VARCHAR(10) NOT NULL, -- Must be 'ERROR' in this table.

AZURE\_FUNCTION VARCHAR(5) NOT NULL, -- 'F1'or 'F2'

CORRELATION\_ID VARCHAR(100) NOT NULL, --Facets Message ID (Batch/Group Identifier)

AZURE\_FUNCTION\_RUN\_ID VARCHAR(100) NOT NULL,

-- Event/Error Details

PROCESS\_STEP VARCHAR(50) NOT NULL,

ERROR\_CODE VARCHAR(10), -- Custom Error Code (e.g., F2004).

QUEUE\_MESSAGE VARCHAR, -- Detailed log message / Exception stack trace

-- Retry Context (Used primarily for ERROR logs)

DATA\_URL VARCHAR(500), -- URL to the full claim payload (for F2 retries). Populated by both F1 (Batch errors) and F2 (All errors).

CLAIM\_ID VARCHAR(50), -- Specific Claim ID that failed (if known). NULL for F1 errors and F2 batch-level errors.

CLAIM\_LINE\_NUMBER INTEGER, -- Specific Claim Line that failed (if known). NULL for F1 errors and F2 batch-level errors.

RETRY\_COUNT INTEGER DEFAULT 0,

IS\_RESOLVED BOOLEAN DEFAULT FALSE, --Set to TRUE upon successful reprocessing

-- Raw Data (For deep investigation of F2004 type errors)

RAW\_INPUT\_DATA VARCHAR -- Small chunk of input data relevant to the failure (e.g., the specific claim JSON object that failed validation)

);

# 5. Amendment History

| **Version** | **Date** | **Description of Change** | **Prepared / Revised By** |
| --- | --- | --- | --- |
| 1.0 | 10/06/2025 | Initial creation of System Design Document based on SRS requirements, defining the Azure function Pipeline, Dataflow, logging strategy, and Snowflake schema. Subsequent updates added detail Audit and Error logging tables with full DDL scripts, refined the logging strategy to capture only ERROR logs for retry purposes, clarified logging rules for batch vs claim-level retries, enriched BlobProcessSnowflakeData with external Member API integration and new error handling rules, and corrected formatting and completeness issues in key section and scripts. | Saikiran Reddy P |