**Document Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable Title:** | IFRS 17 Solution: Data Architecture document | | |
| **Owner:** | Subodh Kumar Ojha, Angelo Tang | **Deliverable ID:** |  |
| **Deloitte Reviewer:** | Ross Moore | **Review Date:** |  |
| **Prudential Reviewer:** | William YP Chan , Benny Ho | **Review Date:** |  |
| **Prudential Approver:** |  | **Sign-Off Date:** |  |

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Version Development** | **Author** |
| 18-Oct-19 | template | * Initial template based | Janice Li |
| 26-Nov-19 | Draft | * Restructuring and updated TOC * Added Data Architecture & Data Flow , Data quality * Added Data Modeling Approach, Design Principle | Subodh Ojha |
| 06-Dec--2019 | 0.4 | * Target Data flow architecture * Added Data &ETL dependencies | Subodh Ojha  Angelo Tang |
| 13-Dec-2019 | 0.5 | * TOC updated based on initial review comments * Updated other sections | Subodh Ojha |
| 24-Dec-2019 | 0.5 | * Updated Solution overview and other sections * ETL dependencies | Subodh Ojha  Angelo Tang |
| 31-Dec-2019 | 0.6 | * Reorganized and added ABC framework closed to Based version * Re-Formatting and reviewed and updated Design principles | Subodh Ojha  Manash Nayak |

|  |  |  |  |
| --- | --- | --- | --- |
| 10-Jan-2010 | 1.0 | * Incorporates review comments from Eric and Ross * Added ETL Dependencies Diagram and fine tune | Subodh Ojha |
| 05-Feb-2010 | 1.1 | * Design changes due to Infrastructure cost optimization as LBU specific DWH instance to Single DWH instance | Subodh Ojha |
| 11-Feb-2010 | 1.2 | * Schema design updated due to HK & MY Data Privacy/Security requirement * SAS Schema included in DHW | Subodh Ojha |

**Prudential Life IFRS 17**

**IFRS 17 Solution: Data Architecture**

Table of Contents

[1.0 Introduction 4](#_Toc31793068)

[1.1 Purpose 4](#_Toc31793069)

[1.2 Scope 4](#_Toc31793070)

[1.3 Out of Scope 6](#_Toc31793071)

[2.0 IFRS17 Data Solution Architecture & Data Flows 7](#_Toc31793072)

[2.1.1 IFRS17 Data Solution Architecture 8](#_Toc31793073)

[2.1.2 Azure Data Stores and Data Flow 9](#_Toc31793074)

[2.1.3 Regional Vs LBU specific Solution Overview 10](#_Toc31793075)

[2.2 Data Warehouse Architecture Design Principles 10](#_Toc31793076)

[2.2.1 Landing Layer: Guiding Principles 13](#_Toc31793077)

[2.2.2 Staging Layer: Guiding Principles 14](#_Toc31793078)

[2.2.3 Abstraction Layer: Guiding Principles 14](#_Toc31793079)

[2.2.4 Foundation Layer: Guiding Principles 14](#_Toc31793080)

[2.2.4 Insight Layer: Guiding Principles 15](#_Toc31793081)

[2.2.5 SQL DW Schema Design 16](#_Toc31793082)

[2.2.6 Master/Reference Data Design Principles: 16](#_Toc31793083)

[2.2.7 Data Validation Design principles 17](#_Toc31793084)

[2.2.8 Data Exceptions Design principles 17](#_Toc31793085)

[2.2.9 Data Reconciliation Design principles 18](#_Toc31793086)

[2.2.10 Data Cleansing Design principles 18](#_Toc31793087)

[2.3 Data Modelling Approach & guiding principles 19](#_Toc31793088)

[2.3.1 Data Model Patterns and guiding principles 19](#_Toc31793089)

[2.3.2 Logical Data modeling guiding principles 20](#_Toc31793090)

[2.3.3 Physical Data modeling guiding principles 20](#_Toc31793091)

[2.3.3.1 Choosing Hash Distribution Key Column for a Hash Distributed Table 20](#_Toc31793092)

[2.3.3.2 SQL Data Warehouse Table Partitioning strategy and recommendation 22](#_Toc31793093)

[2.4 Data Security and Controls Principles 22](#_Toc31793094)

[3.0 IFRS17 Data Sources & Targets systems 23](#_Toc31793096)

[3.1IFRS17 Source systems 23](#_Toc31793097)

[3.1.1 PHKL Source systems for IFRS17 24](#_Toc31793098)

[3.1.2 <LBU2> Source systems for IFRS17 25](#_Toc31793099)

[3.1.3 <LBU3> Source systems for IFRS17 25](#_Toc31793100)

[3.1.4 <LBU4> Source systems for IFRS17 25](#_Toc31793101)

[3.1.5 <LBU5> Source systems for IFRS17 25](#_Toc31793102)

[3.1.6 <LBU6> Source systems for IFRS17 25](#_Toc31793103)

[3.1.7 <LBU7> Source systems for IFRS17 26](#_Toc31793104)

[3.1.8 <LBU8> Source systems for IFRS17 26](#_Toc31793105)

[3.1.9 <LBU9> Source systems for IFRS17 26](#_Toc31793106)

[3.2 IFRS17 Targets systems 26](#_Toc31793107)

[3.2.1 Prophet Data Flow 26](#_Toc31793108)

[3.2.2 SAS Sub Ledger Data Flow 27](#_Toc31793109)

[3.2.2 SUN GL Data Flow 29](#_Toc31793110)

[4.0 Data archival and Data retention 30](#_Toc31793111)

[5.0 Data Ingestion & ETL Approach 31](#_Toc31793112)

[5.1 ETL Framework 31](#_Toc31793113)

[5.2 Data & ETL Solution Dependencies 32](#_Toc31793114)

[5.2.1 Source Extraction Considerations for LBU IT 33](#_Toc31793115)

[5.2.2 Landing Layer Folder Structure recommendation 34](#_Toc31793116)

[5.3 ETL Audit, Balance and Control Framework (ABC Framework) 36](#_Toc31793120)

[5.3.1 ABC Audit Process Flow 38](#_Toc31793121)

[5.3.2 Record Count Check Flow (Landing to Staging) 39](#_Toc31793122)

[5.3.3 Record Count Check Flow (Staging to Foundation) 40](#_Toc31793123)

[5.4 CI/CD & Control-M Approach 44](#_Toc31793124)

[6.0 References 46](#_Toc31793125)

[7.0 Assumptions 46](#_Toc31793126)

# 1.0 Introduction

The Data Architecture document aims to provide Prudential with a standard approach towards data management, aligned with best practices as well as corresponding activities needed to be performed. The objective is to provide guidance to the project team and stakeholders around all the activities that need to be undertaken to successfully extract data from source applications, transform the data through business logic to build the IFRS 17 datasets and load it into the new CSM Engine that will generate the IFRS 17 Postings.

The IFRS17 program provides Prudential with an opportunity to simplify and consolidate their business processes, as well as an opportunity to review, rationalize, and enforce, where required, data quality and standardization.

## 1.1 Purpose

This document provides Prudential with an illustration of the high-level Data Architecture for the IFRS 17 program focusing on:

* How data is Processed, Stored and utilized in the IFRS 17 Solution including the following:
  + Data solution Architecture and end to end Data Flow
  + Data Warehouse guiding principles
  + Data and ETL dependencies considerations
  + Data validation and cleansing considerations
  + Data reconciliation and Audit balance and error handling
* Data Modeling Approaches and guiding principals
* Describe the Data warehouse design principles , Data partitioning , Distribution and indexing strategy
* ETL framework , Data ingestion approaches
* Technical information that will help the Data Architects/ Modeler with the ETL Architects & developers to under Data and ETL Architecture and Design for the IFRS 17 Solution.

Also each ETL work stream will produce the Technical specifications which includes each and every ETL/DB component level design details based on functional specifications.

## 1.2 Scope

The scope of this document is to provide the end-to-end Data Architecture to support the delivery of the IFRS 17 Solution for Prudential. This Data Architecture document, developed with Prudential Hong Kong Limited (PHKL), which defines the guiding principles, that will support the rollout of the IFRS 17 Data & ETL capabilities for Prudential’s Local Business Units (LBU) in the region. This document will include to cover the Data Architecture, Data Flow design, Data Modelling approaches/ principles, ETL dependencies source and targets and data archival and retention policies. Any of the LBU system , PAS or target system like prophet Enterprise, SAS SGL and Sun GL are the out of scope for the document .

Data from Prudential’s current in-scope source systems as defined in Solution Design will be extracted into the IFRS 17 Platform. Additional discovery during the design phase will validate the source systems and identify any additional sources outside the systems listed in the Solution design document.

The scope of the document is limited to IFRS17 end-to-end Data solution Architecture and Data Flow design, Data Modelling approaches, Data Storage design principles to support the Data and ETL capabilities of the IFRS 17 solution. It includes the followings

* + Data Architecture to support the IFRS 17 Data and ETL capabilities of the solution
  + Data Modelling approach and guiding principles
  + Data flows across the ETL Framework layers, within the Azure Cloud Platform, across the data storage component to support the IFRS 17 Solution.
  + Data Ingestion and ETL Approach
  + Data & ETL dependencies
  + CI/CD and Control-M Architecture

Baseline Solution Design was developed with PHKL and reviewed with four LBUs to assess regional applicability

The new IFRS 17 platform, will be deployed in the Azure Cloud platform with Azure Data Lake Store (ADLS) and Azure SQL Data Warehouse (ASDW). As a Pilot Phase and deployed for PHKL, as outlined in the indicative table below.

|  |  |
| --- | --- |
| **IFRS 17 ETL** | **ETL Scope Area** |
| ETL1 | Valuation extracts: Mastering Coverage-level IFRS 17 Portfolio, Combination, Cohort, Contract Boundary |
| ETL2 | Prophet Input Data and Asset Share roll-forward |
| ETL3 | Prophet Outputs (Expected Cash Flow) to CSM Input |
| ETL4 | PAS Accounting Transactions, Control and Offset |
| ETL5 | Other Accounting Transactions, Control and Offset, Allocation and prep for Sub Ledger |
| ETL6 | PAS Actual Cash flows and Unit of Account tagging |
| ETL7 | Sub Ledger Reference Data and Preparation for Sub Ledger |

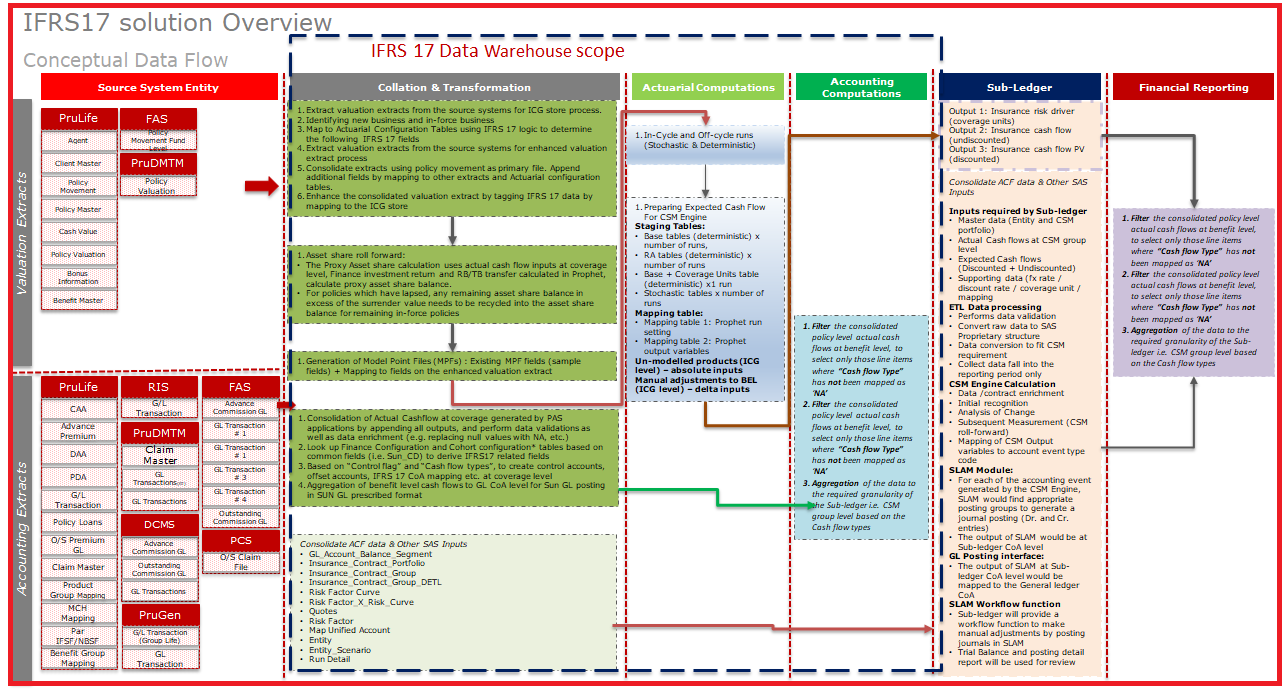
## 1.3 Out of Scope

* The extraction of source data from each of the LBU source systems is the responsibility of the specific LBU IT teams to support the IFRS 17 Solution. Each LBU IT Teams will manage and operate the Source Extraction Program in their respective LBU.
* The LBU IT Teams would be responsible for the data movement of both the source data and control files to the Landing Layer in the Azure Data Lake Gen2. The specific LBU IT Teams will use MoveIT to deliver the data to the Landing Layer.
* Functional specifications document
* ETL spec’s (source to target mapping) to allow build to start
* Data Model and Data dictionary
* Interface agreement document
* Detailed application component design for SAS Sub-ledger
* Detailed application component design for Sun GL
* Detailed infrastructure , Cloud and network , physical database design component for Azure

# 2.0 IFRS17 Data Solution Architecture & Data Flows

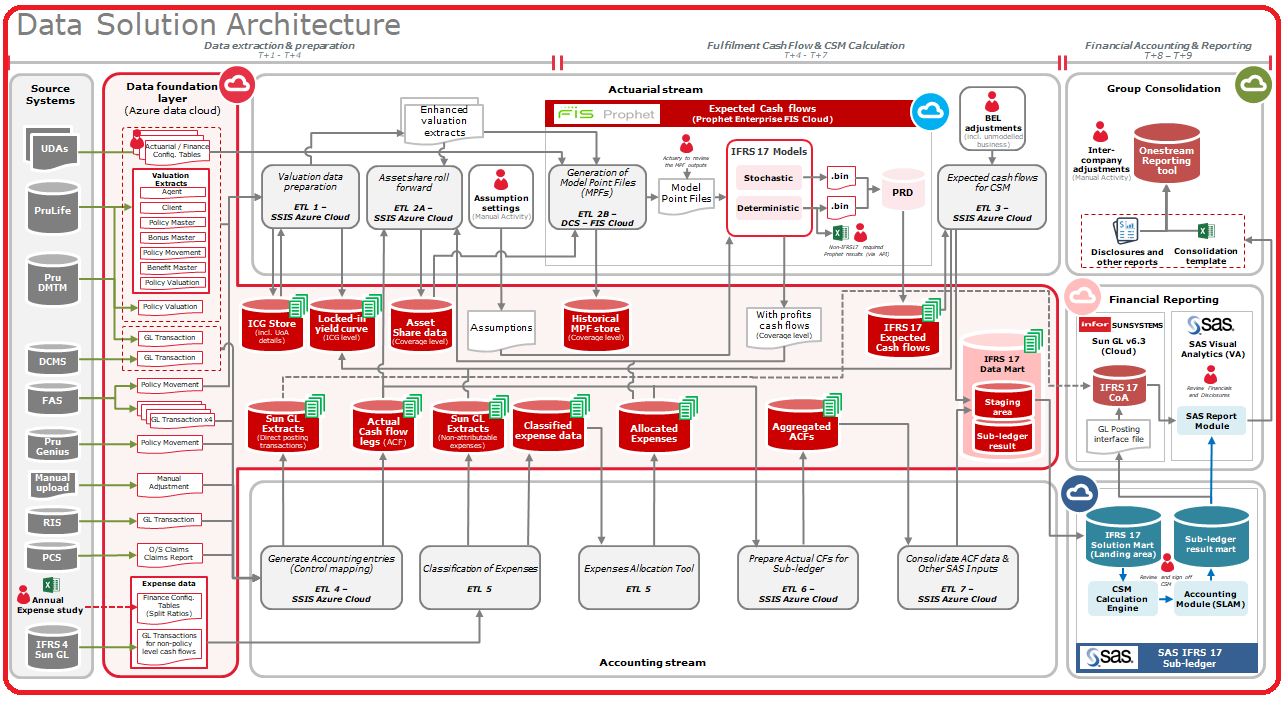
**IFRS17 Conceptual Overview (Functional)**

This below diagram descript how IFRS17 functional flow starting from PAS and other Source systems to IFRS17 GL posting and reporting requirement along with interaction with Actuaries Data (Prophet Cloud), SAS Sub leger and Sun GL



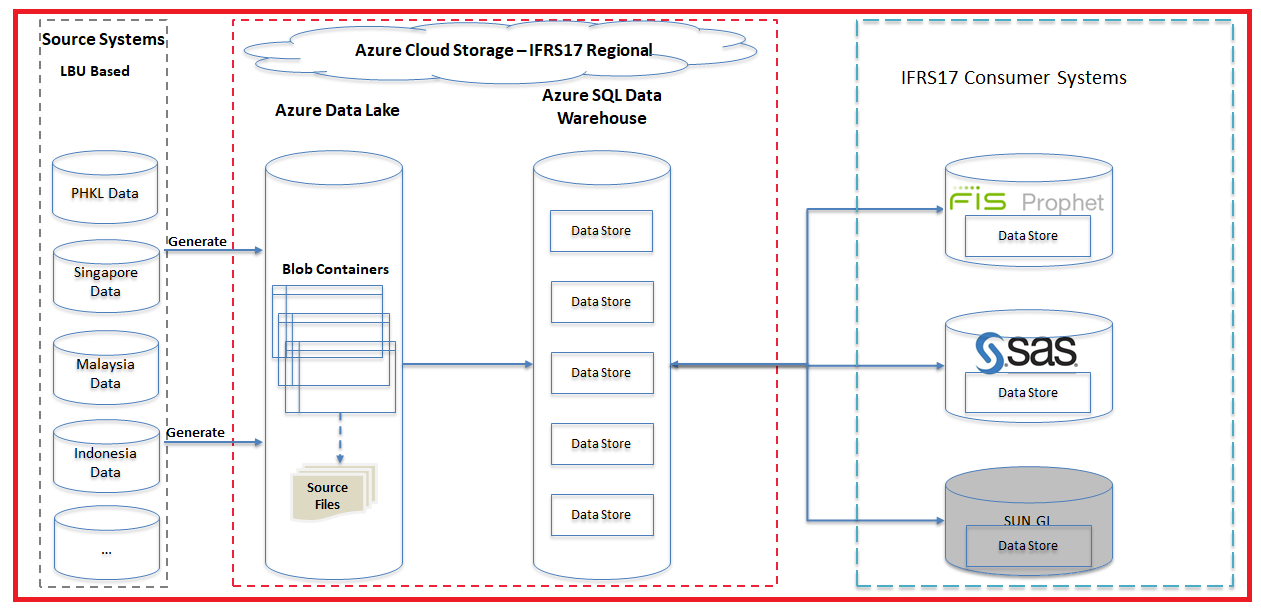
## 2.1.1 IFRS17 Data Solution Architecture

This below diagram descript how IFRS17 data flow starting from Source system to IFRS17 GL posting and reporting along with interaction with Actuaries Data (Prophet Cloud), SAS Sub leger and Sun GL



## 2.1.2 Azure Data Stores and Data Flow

High Level Architecture for the Data Store and Data Flow in the Cloud:



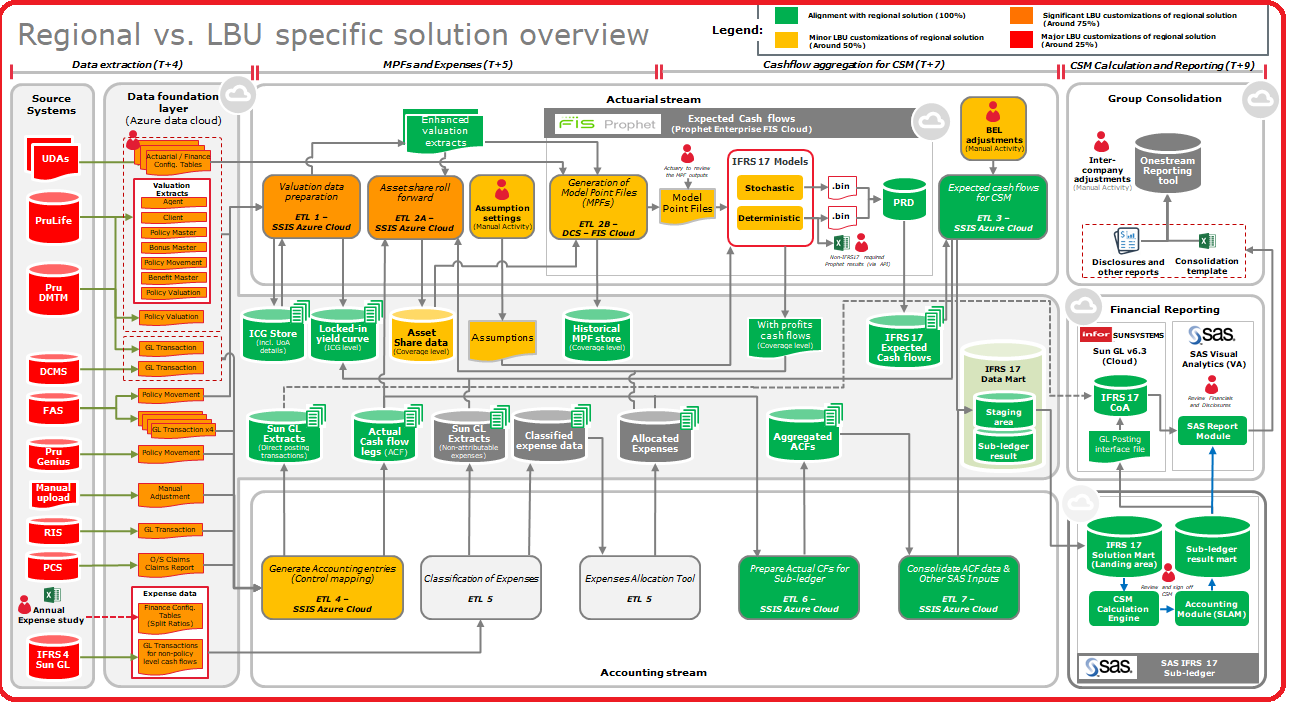
The diagram shows the high level data store architecture in the cloud. The source data files will be generated from different business systems based on different LBUs.

Next, the generated source files will be put into the specified Blob Directory in the specified Blob Container in the Azure Data Lake (ADL) via MoveIT tool. All generated source files, Output/Target files, Master/Reference data files, etc. will be stored in this ADLS.

After that, those source files will be ingested, transformed and loaded into the Azure Data Warehouse via SSIS Packages. And in this process of data processing, the raw data will be processed in multiple data layers for validating, cleansing, unifying, consolidating and aggregating.

Finally, the transformed data or the calculated data will be consumed by downstream systems like Prophet Actuarial Model, SAS CSM Engine and SUN GL Finance system.

## 2.1.3 Regional Vs LBU specific Solution Overview

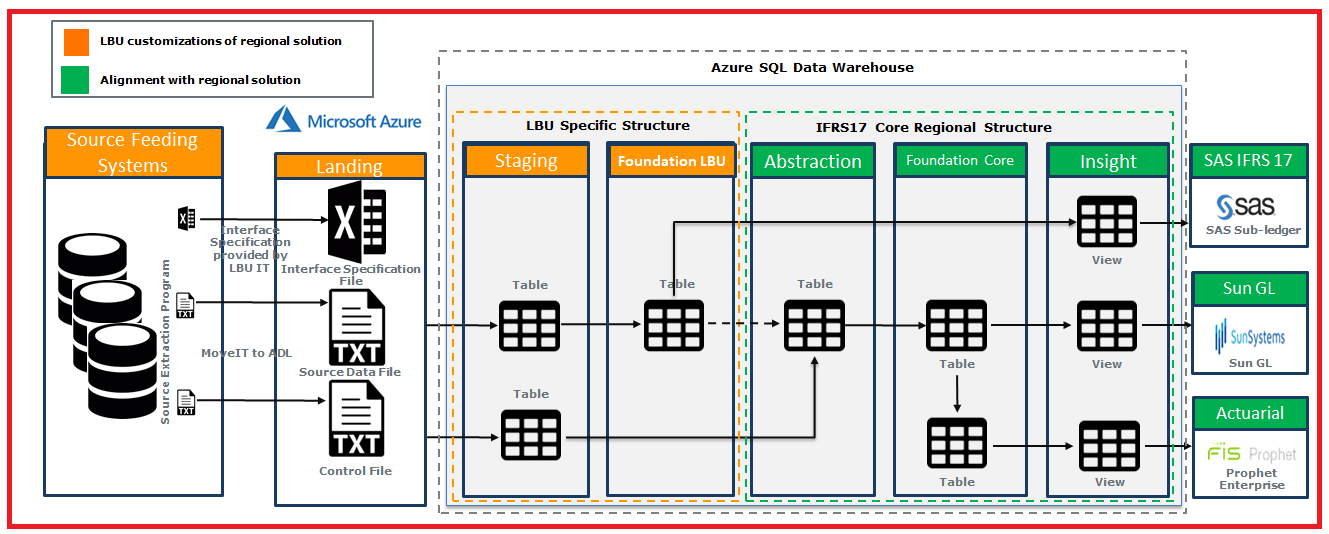


## 2.2 Data Warehouse Architecture Design Principles

IFRS17 data warehouse architecture Design principles are considering in multiple aspects like technology, data requirement, reference data management, how loosely couple with multiple layers and how flexible should be the data model to cater scalable and flexible for further enhancement . Also the data and information must conform to required internal and external security and privacy standards, policies and regulations.

Considering Azure Data Lake Gen2 for Data landing Layer and single Azure SQL DW instance for all LBU for IFRS17 Data warehouse layer with below properties includes detail LBU schema design & naming conventions

IFRS17 Data warehouse has five (5) logical layers (Landing, Staging, Abstraction, Foundation and Insight) within single DW instances with LBU specific DB schemas and DB schema in Azure SQL DW (STAGING, FOND\_LBU, ABST\_LBU, FCORE\_LBU and INST\_LBU) each serving a specific purpose, which supports in delivering a flexible, reusable and scalable data Platform regional implementation. The Color coding links from above Regional vs LBU scope for data.



**Rationale for Landing Layer:**

* The Landing layer is where the solution will store all the data that it receive from the source applications in its raw form. This layer will be from the source in its raw from. Each of the LBU Source Feeding Applications will have Source Extraction Programs generate the data to support the IFRS 17 solution. Each Source Extraction Program will generate one (1) Source Data File and a corresponding Control File. MoveIT or Batch Scripted SSFTP will execute the movement of both, the Data File and Control on Azure Data Lake Gen2. The respective LBUs are responsible for the: 1) Data Extraction from the source systems and; 2) Movement of data from on premise to Azure Data Lake
* Landing layer is where we store all the data that we receive from the LBU source system in its raw form.
* This layer will become the source of truth for the IFRS 17 Solution.
* The LBU IT teams will generate and land the data into the Landing Layer

**Rationale for Staging Layer:**

* This layer serves as a System of Record (SOR) pattern of the source system. This Layer will hold the newly arrived data from landing layer and will be kept before business transformations are to be performed. ETL pipelines ingest data into physical tables from data files (.TXT or .csv) in the Data Lake.
* Staging layer is where the newly arrived will be kept before business transformations are to be performed.
* Support Data Profiling activities at this point to understand the health of the data
* Support activities like data standardization and data de-duplication will be hosted in this layer depends on business requirement and viability of tools.

**Rationale for Abstraction Layer:**

* The Abstraction Layer is a non-persistent layer to combine data across LBUs to create a common structure.  This is to support the mapping of data structures from each of the LBUs to create consistency of data into the core IFRS 17 Solution.
* A transient layer to combine data across LBUs to create a common structure. This is to map into a consistent structure across all LBUs to create consistency.
* This Layer support to regional implementation and bring unified layer for all LBU's and LBU specific requirement

**Rationale for Foundation Layer:**

* Foundation layer is the business layer where data will be massaged, transformed and stored to support wider business audience
* Tables in Foundation layer will either be built from the source (Landing through Staging layers) or from another table from Foundation later itself or from Abstraction Layer.
* Foundation Layer will segregates into 2 logical grouping with separate schemas; Specifically FOND\_LBU proposed for ETL1 and ETL2   .
* Data augmentation is performed in this layer when supporting multiple business needs.
* Data Cleansing, Data Integration and Transformation Logic applied on the data to process the results that create IFRS 17-wide Foundation data assets.

**Rationale for Insight Layer:**

* The Insights Layer decouples the curated business layer from delivery of the data to downstream applications. Generally, the Insights Layer is a view with a one-to-one mapping to a table in the Foundation Layer based on the consumer's requirement. In order to, deliver a specific requirement with needs to support heavy tabulations, a materialized view can be created to deliver the results.
* A materialized view will be created for requirements that will serve a specific requirement with a need for heavy tabulations
* Supports downstream application (Prophet Enterprise, Sun GL and SAS Sub ledger) data consumption pattern.
* This includes the ability to deliver views to support Reporting purposes of the IFRS 17 Solution,

**Data Flow / interactions within Internal ETLs**

* + Any data required from FOND ( Regional core) to FOND\_LBU for LBU specific processing then INSIGHT layer to be used, for example
    - For example : ETL 3 to update **LOCKED\_IN\_RATE**  in FOND\_HKILT( LBU) added by ETL 1, So **LOCKED\_IN\_RATE**  View to be added in INSIGHT layer and ETL 1 will pull the data from INSIGHT layer and update in FOND\_HKILT(LBU)
  + Any data required from FOND\_LBU/STAGING for LBU specific processing then directly use the data from table in STAGING & FOND\_LBU, for example
    - ETL 2 to use the valuation extract from ETL 1- FOND\_HKILT, then ETL 2 can directly access FOND\_HKILT
  + Any data required from FOND\_HKILT(LBU) to ETL must though the ABSTRACTION LAYER
    - For example ETL 7 need the FOND\_HKILT.FOND\_ACT\_CSM\_SUBGRP\_CFG data from ETL1, then ETL7 should request ETL1 provide a table view at ABST layer for ETL7 extraction
  + Any data required from ABST\_XX/FCORE\_XX (regional) for core processing then directly use the data from the table in ABSTRACTION/FCORE, for example
    - ETL 7 needs actual cash flow data from ETL 6, then ETL 7 can directly access FCORE\_XX

**Data flow to other 3rd Party (Prophet Cloud, SAS, and SUN GL) systems must always use data from INSIGHT layer**

### 2.2.1 Landing Layer: Guiding Principles

The Landing Layer, in the respective LBU Blob Storage Containers, support the following principles:

* The Landing Layer on the Azure Data Lake Gen 2, with have an individual Blob Storage Container for each LBU.
* The Landing Layer will retain source data in its raw form for seven (7) years but in Archive Folder
* IFRS 17 ETLs support the ability parameterize each component of the folder path in a streamlined manner to support re-processing. This enables the ETL application to easily pick up and process/re-process data for a: Selected Blob Storage Container, Selected Accounting Period and Selected Application
* There will be Application sub-directories to ensure in the Source Files (SrcFiles) and Archive Files (ArchiveFiles) folders to:
* Streamline the grouping of source data files and control files per source application.
* Eliminate the possibility of duplicate source data file and control files in the Landing Layer.
* The LBUs will leverage existing Source Extraction Programs to extract the data from the Source Feeding Applications. The LBU Source Extraction Programs will generate the source data file and a corresponding control file.
* The LBU Source Extraction Programs will generate both, the Data and Control File in the .TXT format.
* The LBUs are responsible for the generation of the Source Data Files and Control Files from existing Source Extraction Programs.

Where there are data gaps in the Source Extraction Programs, the new requirements will be provided to the respective LBU IT teams to action. The LBU IT teams would either:

Enhance existing Source Extraction Programs or; Create new Source Extraction Programs to extract the data to support the IFRS 17 source data requirements.

### 2.2.2 Staging Layer: Guiding Principles

The Staging Layer, in the Azure SQL Data Warehouse Service, will support the following principles:

* The Staging Layer, deployed as a schema in the LBU specific database instance within the Azure SQL Data Warehouse Service with separate schema for Staging Layer.
* Each LBU will have its own IFRS 17 database instance. This will isolate the LBU specific data within its on database instance.
* The Staging Layer will have its own schema, STAG defined in each of the LBU specific IFRS 17 database instances.
* The data structures in the Staging Layer is will be defined based on the Interface Specification file provide the respective LBU IT teams.
* Data profiling activities will be executed on the data in the Staging Layer to understand the health of the data, example:
  + To validate is every column has a value or if some have blank values.
  + To gain an understanding if there are columns that reference a Reference List of Values and if the Reference Data is consistent with the data values available in the column.
* The Staging Layer supports activities like data enrichment and data de-duplication activities. Where there are requirements from business for data enrichment and data deduplication, it will be implemented in this layer. It ensures data is ready for processing

### 2.2.3 Abstraction Layer: Guiding Principles

The Abstraction Layer, in the Azure SQL Data Warehouse Service, will support the following principles:

* The Abstraction Layer, deployed as a schema in the LBU specific database instance within the Azure SQL Data Warehouse services with separate schema for Abstraction Layer
* Each LBU will have its own IFRS 17 database instance. This will isolate the LBU specific data within its on database instance.
* The Abstraction Layer will have its own schema, ABST defined in each of the LBU specific IFRS 17 database instances.
* The respective LBU IT teams are responsible for mapping and developing the ETL pipelines to ingest and validate the source data into the Abstraction Layer.
* The Abstraction Layer operates as a transient layer of data for the current Account Period.

### 2.2.4 Foundation Layer: Guiding Principles

The Foundation Layer, in the Azure SQL Data Warehouse Service, will support the following principles:

* The Foundation Layer, deployed as a schema in the LBU specific database instance within the Azure SQL Data Warehouse Service with separate schema for Foundation Layer.
* Each LBU will have its own IFRS 17 database instance.  This will isolate the LBU specific data within its on database instance.
* The Foundation Layer will have its own schema, FOND defined in each of the LBU specific IFRS 17 database instances.
* The Foundation Layer, is where data is integrated to provide context and meaning to business.
* The ETL pipeline ingesting the data into this layer will incorporate the following capabilities – Data Cleansing, Data Integration and Transformation Logic on the data to process the data, that creates Prudential IFRS 17-wide Foundation data assets.
* The Foundation Layer enables the integration of the IFRS 17 ETL Applications (ETL 1 – ETL 7).  It creates a common layer of IFRS 17 enterprise data objects and promotes the interaction of data across the solution.
* Foundation Layer will segregates into 2 logical grouping  with separate schemas as below
* Group of tables which is specific to LBU data  and can be stored in separate DB schema ( FOND\_<LBU>) and   table naming convention  like  FOND\_<TableName> so that in future LBU can reuse for other reporting purpose and implement their data privacy /security control on it  .Specifically FOND\_LBU  proposed for ETL1 and ETL2 Rationale for introducing this concept is as follows
  + In PHKL it was observed that for ETL1 and ETL2 pure LBU specific activities were being done which had more data dimensions than what is required for the core ETL1 and ETL2 solution. The reason for this is ETL1 and ETL2 are having following components which differ by each LBU and hence cannot be kept consistent across
    - LBU specific valuation extract
    - LBU specific ETL 2 Model point file
    - LBU specific MI reporting
    - LBU specific asset share calculation
* Hence, by introducing FOND\_LBU we are solving for
* An optional yet, permanent storage to keep LBU Specific data; for future data query purposes – depending on the respective Data requirements
* An easy transformation (a simple 1-1 mapping) between the FOND\_LBU and ABSTRACTION data layer such that it can be fed into the core ETL1 and ETL2 solution
* Group of tables for IFRS17 Core data (regional) storing in separate Foundation CORE DB schema (FCORE\_<LBU>) for each LBU which will be populating from Abstraction layer mostly as regional core solutions. The naming convention should follow the same naming conventions  FOND\_<TableName>

### 2.2.4 Insight Layer: Guiding Principles

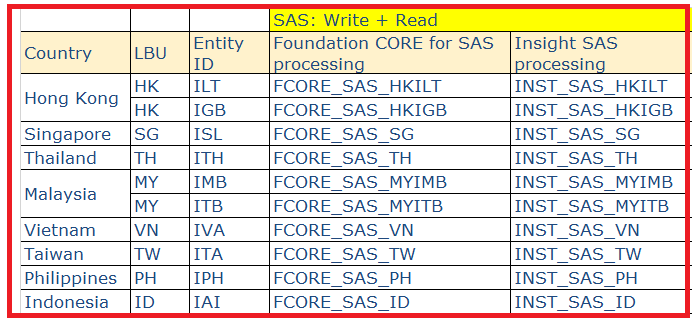
The Insights Layer, in the Azure SQL Data Warehouse Service, will support the following principles:

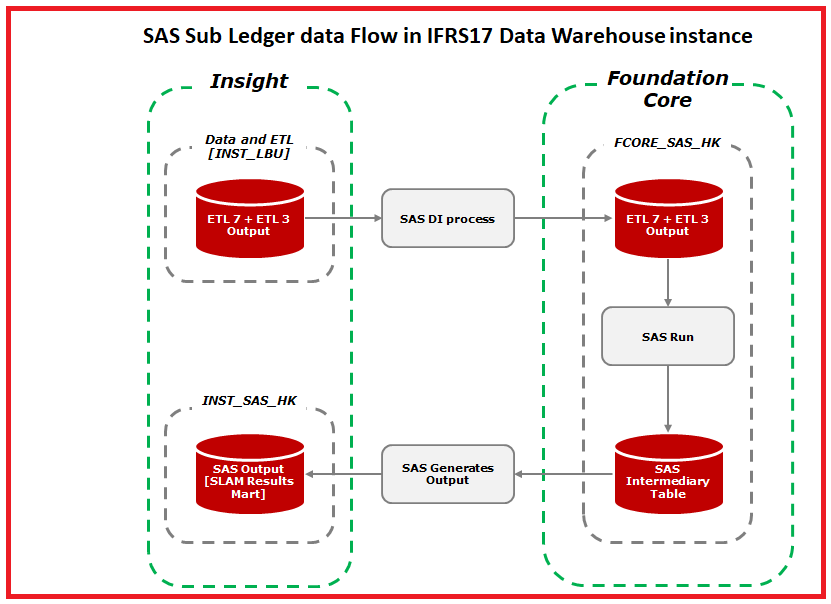
* The Insights Layer, deployed as a schema in the LBU specific database instance within the Azure SQL Data Warehouse Service with separate schema for Insight Later.
* Each LBU will have its own IFRS 17 database instance. This will isolate the LBU specific data within its on database instance.
* The Insights Layer will have its own schema, INST defined in each of the LBU specific IFRS 17 database instances.
* The Insights Layer, supports downstream application (Prophet Enterprise, Sun GL and SAS Sub ledger) data consumption pattern. This includes the ability to deliver views to support MI/Reporting purposes of the IFRS 17 Solution.
* Insights Layer will always be a view with a one to one mapping to a table in the Foundation layer.
* View (or Materialized view) will exposed based on requirements that will serve a specific to consumer (SAS VA / SUN GL) like semantic layer for reporting requirement.
* Materialized View creation should have limited to maximum 5 underlying table joining and not complex calculation or transformation.
* To avoid Materialized view where underlying data sets that are frequently updated.

### 2.2.5 SQL DW Schema Design



The following Schema using by SAS Sub ledger team for their Sub ledger /CSM calculation and use the same IFRS17 Data warehouse.





### 2.2.6 Master/Reference Data Design Principles:

We’re not going to create a data hub for maintaining and distributing master data into core business systems, prefer to identify, maintain master data (or we called reference data) in the azure data warehouse.

The Reference data will be generated manually by users and upload into the specified folders that defined in above folder structure in the landing area in the Azure Data Lake. And we will keep the history reference data in tables in order to satisfy the data reprocessing. For example: previous batch accounting period data.

The Reference data also need to be processed/maintained in the range of specified accounting period.

**Approach**:

* LBU is responsible to generate and upload multiple reference data files into the landing zone in Azure Data Lake.
* The ETL process (SSIS package) will ingest reference data and load into the Azure Data Warehouse.
* Based on user requirements, need to define a set of reference data tables with proper unique key (Business key + Accounting period) to maintain the reference data via Slowly Changing Dimension ( SCD) Type 2 approach as while any update is coming for existing records the SCD type 2 will create new records instead of updating existing ones. The only change that is permitted in an existing record is an update to a columns that indicates whether the record is current or expired and effective from / to dates. Detail design and approaches will be in Technical specification document
* While populating the data into the foundation layer table, source transaction data in the abstraction layer after cleaning and standardizing will lookup maintained reference table to retrieve the reference output result

### 2.2.7 Data Validation Design principles

The challenge for the data team is balancing the time and cost involved in trying to cleanse or connect incomplete or erroneous data.

* Implementing data validation checks within all ETL processes based on exact business requirements mostly NULL check and completeness check.
* Can implement data quality issues based on business provided rules, if require (not impacting the source system).
* Logging data quality issues as exceptions and send to respecting source system via email notifications.

### 2.2.8 Data Exceptions Design principles

There are below types of data exceptions what we are considering

1. **Missing or Unknown Values**

Missing values are the most common data exception because many systems do not put constraints on every data entry field. When you receive missing values, you need to decide whether to replace the value with an “Unknown” identifier or leave the value as blank.

The importance of the column also affects how you handle the data. For example, you may need to flag the record as an exception for manual review. If the value is just an attribute where unknowns are expected, you can use a default value.

Define a default value for the data dimensions in order to ensure that the blank/null fields coming from source are normalized

For e.g. POLICY\_NO coming as blanks/null from PruLife GL transaction table will be normalized as "NA" in the DATAWAREHOUSE

1. **Date-Type Conversion and Out-of-Range Errors**

Another common data problem, especially when dealing with text files, involves conversion problems. This problem manifests itself through truncation errors, range errors with numeric constraint issues, or general conversion issues (such as converting text to numeric).

In this part, the standard is we have to correct the data if it is mandatory to the business analyze. To the non-core KPI, the appropriate solution may involve simply ignoring the value or sending the record to a temporary location until it can be reviewed.

1. **Duplication**

When dealing with duplication of codes and values that are discrete, a lookup mapping table can be maintained to help in the process. Consider managing the mappings through a data validation interface to prevent operations or development from being directly involved.

Two general types of duplication exist:

* Duplication of exact values or records that match across all columns
* Duplication when values are not exact but similar

The suggest solution is the first step in removing duplication is identifying where duplications can exist and the type or range of variations that indicate duplication.

### 2.2.9 Data Reconciliation Design principles

Data reconciliation detail design are specified in ABC framework document and each ETL technical specification document separately. However, here are the high level principles need to follow each ETL application

Exception reports show that the ETL process detected and logged an exception in the Source- (or Extract In-) to-Stage ETL process and send notification to respective team via email.

Flagging Data Exceptions: An alternative approach is to repair the data exception and/or flag the records as exceptions and allow them to flow to the data warehouse.

Reasons you might consider this approach include:

* The data exception is not severe and can be repaired by the ETL process. Missing values and unknown values are examples of this.
* The record has valid data along with the exceptions that need to be flagged.

Following will be validations performed w.r.t. transformation/data flow handling. A full scan pre-check should be completed prior to the loading of data to Staging Layer and respective layers.

* Data Accuracy– Check Sum
* Data Completeness (Applicable to All) – Record Count Check
* In order to ensure the completeness of data copied from Landing to Staging Layer; record count reconciliation would be required to perform.
* Each of the extracts ingested into Data Lake i.e. at Landing Layer would be in a combination of ".csv", ".txt" file and ".ctl" file
* Control file i.e. ".ctl"; will have the information about NO\_OF\_RECORD; after moving the data from Landing to Staging Layer, record count check will be performed in STAG\_<TABLE\_NAME> to ensure the record count matches with the value of NO\_OF\_RECORD
* Reconciliation Report
* Prepare the reconciliation report in below structure; to cater the data completeness and data accuracy checks

### 2.2.10 Data Cleansing Design principles

Data cleansing is the activity of dealing with the data quality issues and data exceptions in the ETL process. Cleansing can range from simply replacing a blank or NULL value to a complicated matching process or de-duplication task. Here are some general guidelines when implementing data cleansing:

1. Multiple layers of data cleansing are often applied in the ETL process. The first layer usually involves data parsing and handling of common data exceptions such as NULLs, unknowns, and data range problems. The second level may involve de-duplication or data correlation to a different source, data consolidation of different sources, or data aggregation and summary.
2. A better approach is to either apply the initial layer of data cleansing in the query that is run against the extract table or leverage the transformation capabilities in the ETL tool. Both of these strategies have much less overhead and allow more server resources and time to be applied to any complicated data cleansing.

Of course, you can also use T-SQL queries to cleanse data, use ISNULL, NULLIF, RTRIM, LTRIM, REPLACE, etc.

When using SSIS for data cleansing, you can apply many of the transformations to different data cleansing situations. Here are some examples:

* You can use the Derived Column transformation for text parsing, NULL identification and replacement, calculations, and more.
* Lookup and Merger Join transformations can help with data correlation and association.

## 2.3 Data Modelling Approach & guiding principles

**Background**

The requirement of IFRS17 is mostly reporting from the subject area includes ; Policy , Agents commission , Claim , benefit, coverage , Cash in-Flows, Cash out-flow , Sub GL , and SUN GL which is coming from multiple policy admin systems and GL and messaging with actuaries & Sub ledger for risk modeling ,CSM calculation and then finally posting to SUN GL as output into IFRS17 reporting/ visualization .

IFRS17 is not creating an enterprise data warehouse in which master policy information, agent information, claim header etc. Also based on business understanding and requirement analysis it is not exactly analytical requirement nor pure operational reporting. So we are propose to use hybrid Model approach of 3NF and Dimensional Modelling. We are capturing the information necessary for the sub ledger and storing it in a structured way, permanently. This still needs a logical and physical data model, it's not a full star schema model because those are designed around reporting requirements.

**Recommendations:**

A good data model is blueprint for any application and data model design plays a vital role in overall application performance & maintainability. It is important for development teams to design, validate and review the data model to match application’s functional and non-functional requirements while using data modeling guiding principles

A conceptual data model is a high level data model identifying entities and their relationships in business context. It is preliminary in structure, mostly abstract in content and sparse in attributes, that is intended to represent a business area

Currently we will start the data modeling from Reverse Engineering approaches from PDM of the existing pilot /Foundation model and continue to enrich and enhance model until all business requirement the we will deliver the complete LDM . However we will try to follow the below modeling approaches based on guiding principles one it will applicable

### 2.3.1 Data Model Patterns and guiding principles

* + We propose to Use of object modeling concepts to group entities (subtypes) that share similar characteristics or behaviors under a common parent (supertype)
  + Supertype and subtype hierarchies accommodate varying levels of abstraction and specificity
  + A supertype is the highest level of abstraction for a business concept – the most conceptualized level
  + The leaf-level subtype (lowest level subtype in a hierarchy) is the lowest level of abstraction for a business concept – the most specific and least conceptualized
  + Relationships are handled using generic associative entities which relate supertype entities (subtype entities do not participate in these relationships)
  + Relies on typing of entities using a type attribute for supertype / subtype hierarchies and subtypes (natures) of relationship (associative) entities

### 2.3.2 Logical Data modeling guiding principles

The components of a logical data model include Entities, Relationships, and Attributes. Each Entity represents a set of persons, things, or concepts about which the business needs information. Each Relationship represents an association between two entities. Each Attribute is a characteristic or piece of information that further describes an entity. A name and a textual definition describe each of these components. These name and definitions provide ongoing documentation of the business rules and information requirements of the business area.

The logical model must:

* + define the entities and attributes, as well as show the relationships that exist between the entities
  + illustrate all sub-types & super-types
  + resolve many-to-many relationships
  + should not use Database specific reserve words

### 2.3.3 Physical Data modeling guiding principles

Since the physical model is technology specific, certain standards and guidelines are mentioned with reference to technology. We are not using LDM to PDM conversion in this scenario so all below rules may not be applicable. However we are recommending in case we can use to enrich the Data Model in future.

Musts for conversion from logical to physical model

* + Subtypes and super-types need to be resolved by either:
    - rolling up subtypes to super-types or,
    - rolling down super-types to subtypes
  + De-normalization may only takes place for performance benefits
  + All entities must translate to tables
  + All attributes in an entity must translate to columns

It’s not mandatory that all logical entities and attributes will be converted as is to a physical tables/columns in addition to 1 to 1 conversion of logical to physical objects , there may be many occurrences of physical only table or there may be occurrences of physical only attributes required as per business requirements. For example rolling up and rolling down of super types and sub types creates physical only tables.

## 2.3.3.1 Choosing Hash Distribution Key Column for a Hash Distributed Table

Choosing the right distribution key column on which data should be distributed across different distributions plays a very important role. When we design our hash distributed table, there are some important considerations that need to take while choosing the distribution key column:

**Data distribution** – We need to ensure that the column that chosen for distribution key has a maximum or higher number of unique values. This is to ensure your data gets distributed evenly across all the compute nodes in order to have a good query parallelization across compute nodes. Contrary to that if you choose a column which has values unequally distributed, it will have data skew-ness. For example, if there are high number of NULL values for the distribution key column, one compute node might end up processing all the data (as all those NULL values will end in a single distribution) and will become bottlenecked in the parallel query performance.

**Compatible Joins** - Data movement over the network is one of the most expensive operations and this is where compatible joins might help. With the help of compatible joins, need to ensure we have all the data needed to process the portion of a query on the local compute node. To achieve this high performance, whenever possible we need to design tables and queries in a way it does not require data movement. SQL Data Warehouse can achieve high performance by performing joins in parallel on the compute nodes and then combining the results on the control node. To improve query performance, tables and queries need to be designed so that, whenever possible, it is not necessary to move data (or minimize data movement at least) prior to the join. For example, one table is distributed on Policy Number and other one is distributed on Transaction Date.

Guidance to Choose Between Hash Distributed Vs. Round-Robin Distributed

These are some best practices and guidance, which will help to decide the type of distribution for tables in SQL Data Warehouse:

|  |  |
| --- | --- |
| **Hash Distribution** | **Round-Robin Distribution** |
| The selection of the distribution key column is done by taking into consideration certain factors like minimizing the data skewness, minimizing the data movement for most of the joins, and types of queries executed on the system. | Usually common dimension tables or tables that don’t distribute evenly or smaller lookup tables are good candidates for a round-robin distributed table. |
| A null able column is a bad candidate for any hash distributed table; likewise, an updated column to be used as a distribution key column is not a good fit. | When the table is a temporary staging table or there is no obvious joining key then a round-robin table makes sense. |
| A column with default constraints is usually not a good candidate for a distribution key column of a hash distributed table as it will introduce data skewness. | When the table does not share a common join key with other tables or the join (in the query which you are optimizing) is less significant than other joins in other queries, again a round-robin table is to go for. |
| The hash distribution key should contain significant distinct values for evenly distribution across. | Round-robin tables typically provide balanced execution. This is because the data is stored evenly across the distributions. |

**Recommendations:** To use Hash Distribution on **Policy Number** which ever tables contains Policy Number and use Round robin in case of significant null values for that particular column .

## 2.3.3.2 SQL Data Warehouse Table Partitioning strategy and recommendation

Table partitions enables to divide DW data into smaller groups of data. In most cases, table partitions are created on a date column. Partitioning is supported on all SQL Data Warehouse table types; including clustered column store, clustered index, and heap. Partitioning is also supported on all distribution types, including both hash and round robin distributed.

Partitioning can benefit data maintenance and query performance. Whether it benefits both or just one is dependent on how data is loaded and whether the same column can be used for both purposes, since partitioning can only be done on one column.

**Benefits to queries:**

Partitioning can also be used to improve query performance. A query that applies a filter to partitioned data can limit the scan to only the qualifying partitions. This method of filtering can avoid a full table scan and only scan a smaller subset of data.

We strongly recommended to partition any big table ( minimum 5 million record per loading) like Accounting period transaction tables is partitioned into 120 ( 10X12) months using the Accounting period field / proper Partition Key design based on situation, then queries that filter on that partition key column can skip searching in partitions that don’t match the filter.

**Sizing partitions**

When creating partitions on clustered column store tables, it is important to consider how many rows belong to each partition. For optimal compression and performance of clustered column store tables, a minimum of 1 million rows per distribution and partition is needed. Before partitions are created, SQL Data Warehouse already divides each table into 60 distributed Colum store. Any partitioning added to a table is in addition to the distributions created behind the scenes. If a table contains fewer than the recommended minimum number of rows per partition, consider using fewer partitions in order to increase the number of rows per partition

**Recommendations for Partitioning**

* Identify Huge Volume Tables which minimum 5 million record need to process in each loading cycle
* Identify the Partition Key (for ex Accounting Period) which we can vary table to table based on Data consumption and consumer's requirement
* Create minimum 10 years Partition while creating table first time (in case of accounting period 12 X 10 partition) with right range and Partition Key (Accounting period YYYYMMM) partition boundaries
* While partitioning can be used to improve performance some scenarios, creating a table with too many partitions can hurt performance under some circumstances. These concerns are especially true for clustered columnstore tables. For partitioning to be helpful, it is important to understand when to use partitioning and the number of partitions to create. There is no hard fast rule as to how many partitions are too many, it depends on data and how many partitions loading simultaneously. A successful partitioning scheme usually has tens to hundreds of partitions, not thousands.

## 2.4 Data Security and Controls Principles

Security is a top concern for managing databases, and it has always been a priority for [Azure SQL Database](https://docs.microsoft.com/en-us/azure/sql-database/index). Also databases can be tightly secured to help satisfy most regulatory or security requirements, including HIPAA, ISO 27001/27002, and PCI DSS Level 1. A current list of security compliance certifications is available at the [Microsoft Trust Center site](https://azure.microsoft.com/support/trust-center/services/). LBU also can choose to place their databases in specific Azure datacenters based on regulatory requirement. SQL Database supports two types of authentication, SQL Server authentication and Azure AD authentication. We recommend the use of Azure AD authentication over the use of SQL Server authentication.

Azure Data Lake Storage Gen2 implements an access control model that supports both Azure role-based access control (RBAC) and access control lists (ACLs). Role-based access control (RBAC) uses role assignments to effectively apply sets of permissions to *security principals*. A *security principal* is an object that represents a user, group, service principal, or managed identity that is defined in Azure Active Directory (AD) that is requesting access to Azure resources.

* Secure Database first
* Use firewall rules to restrict database access
* Enable database authentication
* Azure Active Directory (AD) authentication
* Role-based access control
* Shared Key and Shared Access Signature (SAS) authentication
* Access control lists on files and directories
* Set file and directory level permissions by using access control lists

As a general guideline when securing your Data Warehouse in Azure you would follow the same security best practices in the cloud as you would on-premises.

* Restrict IP addresses which can connect to the Azure Data Warehouse through DW Server Firewall
* Use Windows Authentication where possible, using domain-based accounts will allow you to enforce password complexity, password expiry and more centralized account and permission management.
* Implement Database level security though management of permissions with Custom Roles allowing you to specify explicit permissions at object level or Built in Roles
* When using SQL Server Authentication use complex passwords and assign explicit permissions to objects to reduce risk at a data level.
* Review the following article for guidelines and information on Logins and Accounts within Azure Data Warehouse

# 3.0 IFRS17 Data Sources & Targets systems

## 3.1IFRS17 Source systems

Data from Prudential’s current in-scope source systems as defined in Solution Design will be extracted into the IFRS 17 Platform. Additional discovery during the LBU specific design phase will validate the source systems and identify any additional sources outside the systems listed in the document.

The Data source will vary based on LBU the current scope for PHKL as below and will enrich the document based on requirement from other LBU's

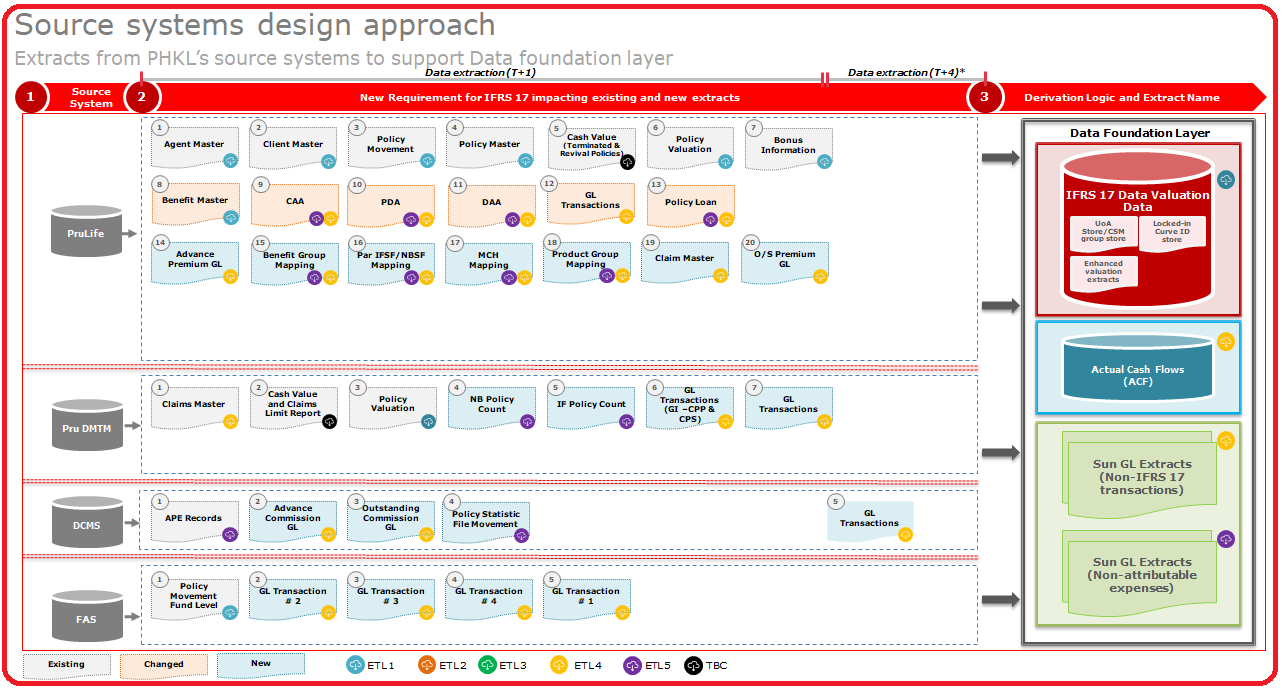
By each LBU/Country, they will generate the source files in each specified Blob directory for each specified Blob Container in the Azure Data Lake. The source files generation according to below rules.

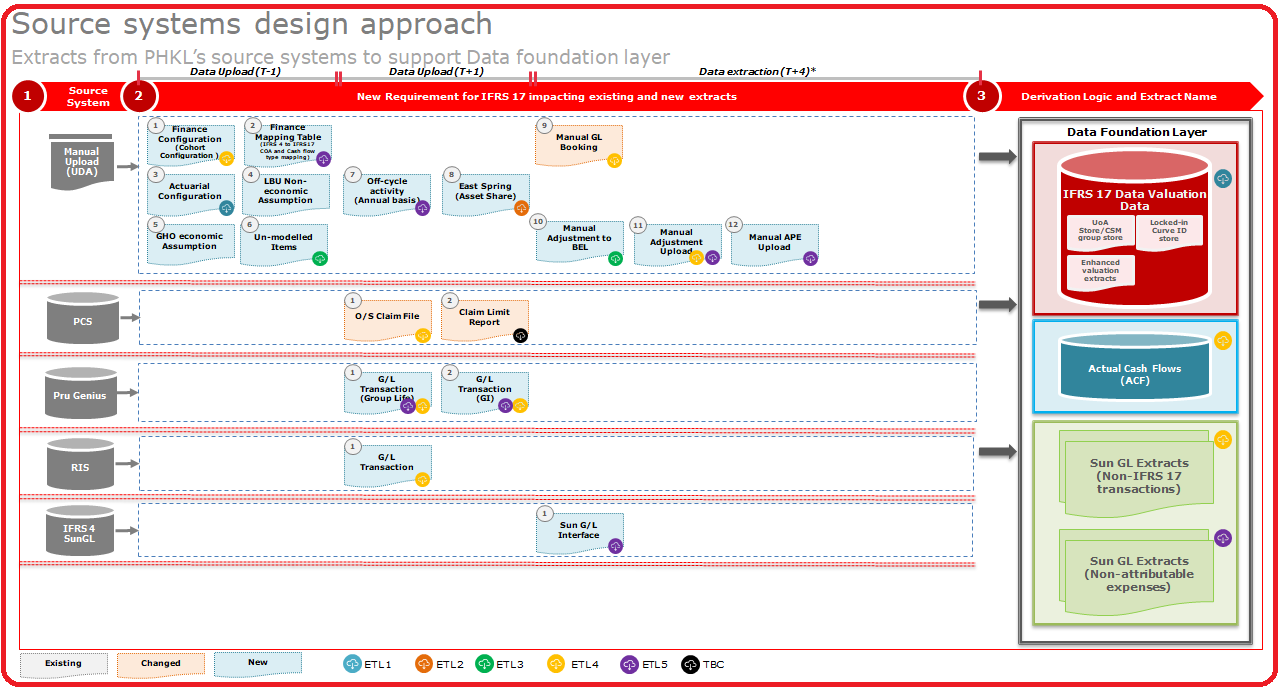
* Extracts from the source system would be copied to a common location by respective applications
* Push mechanism would be used to copy the files from source system to a common area (shared folder).
* MoveIT would be the tool used for copying of files from common area to data foundation layer. Refer to ETL 1 and 4 for detail logic.
* Control checks would be put in place to trace the records selected and extracted from source system are correct. Checksum/MD5 would to be used to check the file is intact while transferring
* Configuration and settings like mapping files would be copied in entirety. However, master file like Policy, Agent, Client, GL etc. would be copied on incremental basis
* In the Azure Data Lake, Blob Container will be created by each LBU/Country and grant access privilege by container level.
* The master/reference source container is a separated one which all LBUs can access their sub-directory and generate master/reference data files.
* Each source file has a corresponding Control File. The control file will record the several key data fields for ETL SSIS packages processing.

For more details please refer to the Control File Specification and Interface file Specification documents.

Each LBU has its own Blob container in the Data Lake and the access control will be set in the Container level.

## 3.1.1 PHKL Source systems for IFRS17





## 3.1.2 <LBU2> Source systems for IFRS17

< Place holder LBUs>

## 3.1.3 <LBU3> Source systems for IFRS17

< Place holder LBUs>

## 3.1.4 <LBU4> Source systems for IFRS17

< Place holder LBUs>

## 3.1.5 <LBU5> Source systems for IFRS17

< Place holder LBUs>

## 3.1.6 <LBU6> Source systems for IFRS17

< Place holder LBUs>

## 3.1.7 <LBU7> Source systems for IFRS17

< Place holder LBUs>

## 3.1.8 <LBU8> Source systems for IFRS17

< Place holder LBUs>

## 3.1.9 <LBU9> Source systems for IFRS17

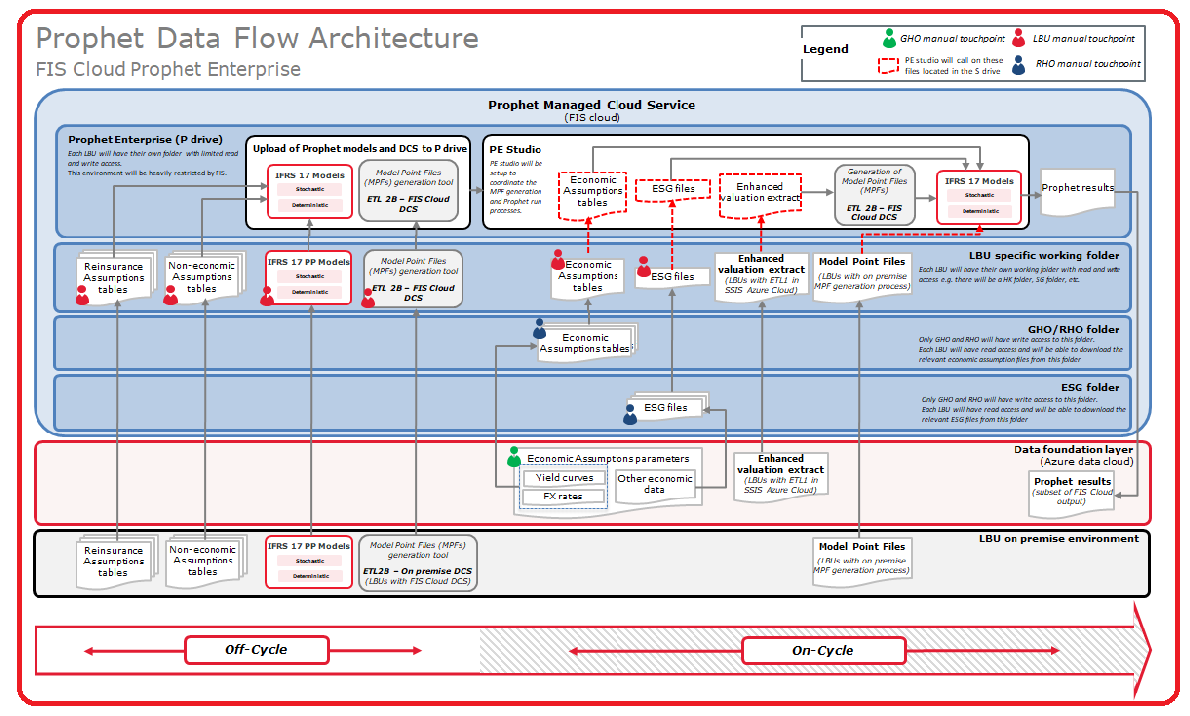
< Place holder LBUs>

## 3.2 IFRS17 Targets systems

In IFRS17 is a very complex calculation with multiple source and target systems outside of our Data warehouse. IFRS17 will flow the data to mainly 3 target Prophet FIS Cloud, SAS Sub Ledger , Sun GL and finally to SAS VA for reporting . The scope of the document is not describes the target systems rather high level data flow overview of 3 target systems

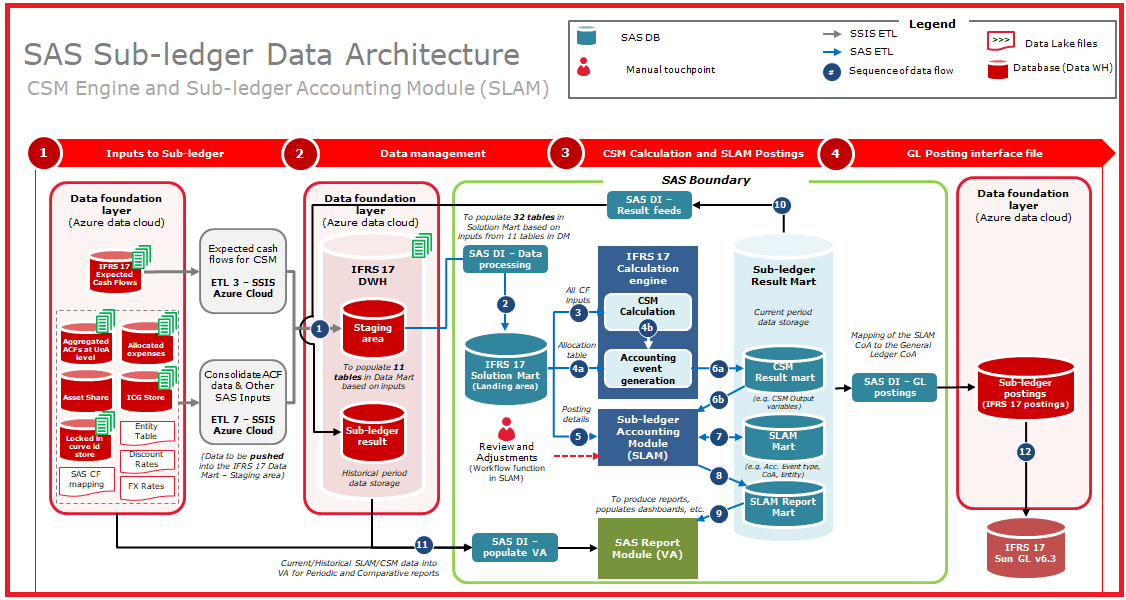
### 3.2.1 Prophet Data Flow

The below Diagram describe data flow between IFRS17 DW and Prophet Enterprise. As per scope of this document will not cover details of Prophet Cloud data flow. The technical details will cover in Prophet Architecture documents.



### 3.2.2 SAS Sub Ledger Data Flow

The below Diagram describe data flow between IFRS17 DW and SAS Sub-Ledger system. As per scope of this document will not cover details of Prophet Cloud data flow. The technical details will cover in SAS Sub-Ledger Architecture documents.



***Inputs required by Sub-ledger***

* *Master/Reference data (Entity and CSM portfolio)*
* *Actual Cash flows at CSM group level*
* *Expected Cash flows (Discounted + Undiscounted)*
* *Supporting data (FX rate / discount rate / coverage unit / mapping*

***ETL Data processing***

* *Performs data validation*
* *Convert raw data to SAS Proprietary structure*
* *Data conversion to fit CSM requirement*
* *Collect data fall into the reporting period only*

***CSM Engine Calculation***

* *Data /contract enrichment*
* *Initial recognition*
* *Analysis of Change*
* *Subsequent Measurement (CSM roll-forward)*
* *Mapping of CSM Output variables to account event type code*

***SLAM Module:***

* *For each of the accounting event generated by the CSM Engine, SLAM would find appropriate posting groups to generate a journal posting (Dr. and Cr. entries)*
* *The output of SLAM would be at Sub-ledger CoA level*

***GL Posting interface:***

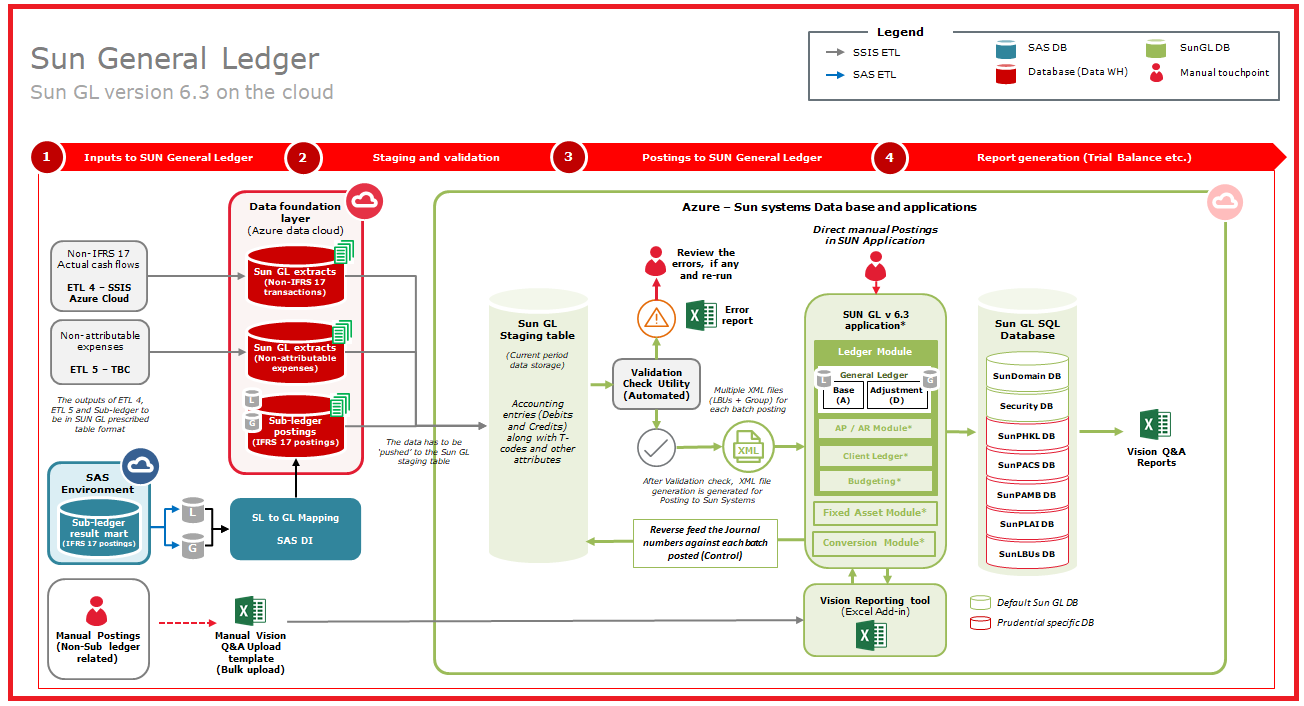
* *The output of SLAM at Sub-ledger CoA level would be mapped to the General ledger CoA*

***SLAM Workflow function***

* *Sub-ledger will provide a workflow function to make manual adjustments by posting journals in SLAM*
* *Manual Adjustments have an auto reversing option*
* *Trial Balance and posting detail report will be used for review*
* *User can drill down to the temporary journal at transaction level*

### 3.2.2 SUN GL Data Flow

The below Diagram describe data flow between IFRS17 DW and SUN GL system. As per scope of this document will not cover details of Prophet Cloud data flow. The technical details will cover in SUN GL Architecture documents.



# 4.0 Data archival and Data retention

Data Lake will store all source system data:

* + There will be a Current Period folder (folder structure design) that will always have the latest period’s source data files by application.  Data is made available on a monthly basis in the Landing Layer.
  + There is a going to be an Archive folder by Accounting Period and Source Application. This layer should retain data for 7 years in its raw format (cold data archive).  The data will be in the archive location will be compressed.  This will support the reprocessing of historical data.
  + However 7 years is not fixed or "hard coded" , ETL will handle such a way that can be done parameterized based on LBU requirement

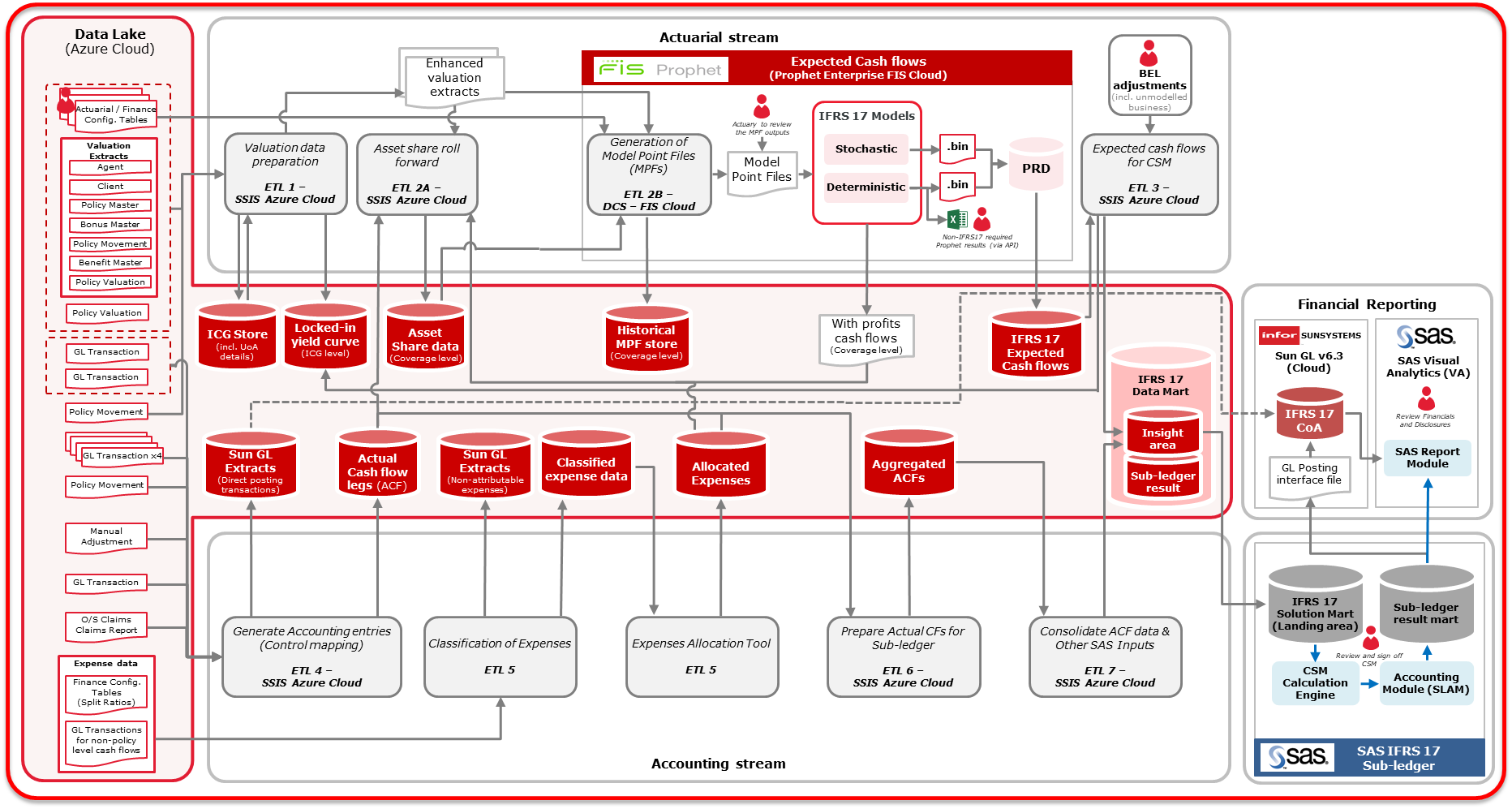
For SQL Data warehouse layer:

* + The Staging Layer has been planned to retain 2 years of data in the data warehouse so, this is the hot retention to support any immediate reprocessing within the data warehouse. The behind that is 2 years data is sufficient for immediate reprocessing and get performance benefits for ETL process. Also in cases of older than 2 years data reprocessing require then raw data can bring from landing archive location ( cold data Archive Layer)
  + The Foundation Layer has been planned to retain 7 years of processed results as part of the IFRS 17 solution.  The same applies to the Insights Layer. More specifically, for inactive policies we have to store 7 years data and In-Forced policies we need to store all data.
  + However 2 years ( in Staging) or 7 years ( in Foundation layer) is not fixed or "hard coded" , ETL will handle such a way that can be done parameterized based on LBU requirement

# 5.0 Data Ingestion & ETL Approach

IFRS17 ETL is very complex and closely couple with multiple systems and other 3rd party system like Prophet Cloud, SAS Sub Ledger and Sun GL system. So we have segregated into 7 ETL stream based on functionalities as specified in scope section.

The below is the high level ETL1 to ETL7 and inter dependency between ETL1 and ETL7



Each of the Local Business Unit’s (LBU) Information Technology (IT) team is responsible for landing both the Source Data and Control File in the Microsoft Azure environment. The LBU-IT team will be extracting the data from the respective LBU Source Application via Source Extraction Programs. Each Source Extraction Program will produce two (2) files:

* Source Data File; and
* Control File

Subsequently, the LBU-IT teams will be move the data into the Source layer on Microsoft Azure. The ETL Application will utilized the data, apply the business logic on it and deliver the data for consumption by the Target applications to deliver the outcomes of the IFRS 17 Solution.

The details around the Source Feeding Systems landing the data into the Source is covered in a subsequent section of this document

### 5.1 ETL Framework

To fuel better collaboration amongst the data and business functions, by enabling and empowering the different teams, and in support of team spirit, through transformed, cleansed and standardized data providing uniformed and synchronized insights to support the IFRS 17 Solution.

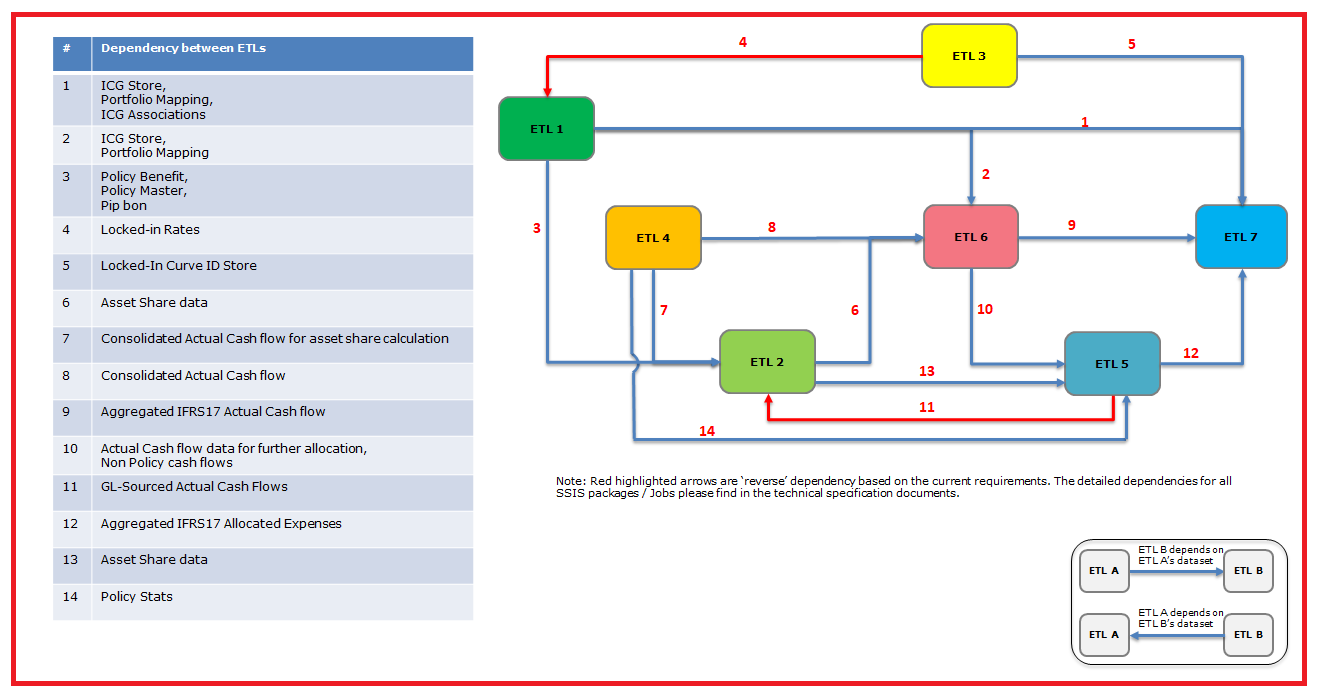
**The Data and ETL Strategy will support:**

* Data consistency and standardisation across data, business functions and systems.
* Enablement of production of accurate and informative reports and alerts to support business information needs and decision making
* Enablement of the organisation to meet future requirements with stable and scalable solutions
* Identification of the primary systems of record from which to acquire source data with support from business users and application engineers

**The Data Conversion Approach outlines the key activities including (but not limited to):**

* Identification of source and target systems (including non-system sources)
* Design of the overall process for ETL development.
* Identification of accountability for data conversion activities (including data profiling and cleansing)
* Definition of ETL (Extract, Transform, Load) tools and platform to be used
* Discuss the data validation activities (i.e. technical reconciliation and business validation)
* Definition of the technical architecture for ETL.
* Describe the high-level plan for Data and ETL by ETL application.

### 5.2 Data & ETL Solution Dependencies



The data & ETL dependencies will support the following principles:

* The Foundation Layer enables the integration of the IFRS 17 ETL Applications (ETL 1 – ETL 7). It creates a common layer of IFRS 17 enterprise data objects and promotes the interaction of data across the solution.
* The Insights Layer, supports downstream application (Prophet Enterprise, Sun GL and SAS Sub ledger) data consumption pattern. This includes the ability to deliver views to support external downstream applications or Reporting purposes of the IFRS 17 Solution.
* The approach of data consumption between the ETL application and the external application use either 'pull' or 'push' operation for the data, that means the external application pull data from the azure data warehouse by using their data extraction tool OR ETL application push the data into the database of the external application by using SSIS.

The ETL Framework provides the foundational approach to develop the ETL components to support the delivery of the IFRS 17 Solution. The ETL Framework based on methodologies that use lean and agile guiding principles to accelerate the data solution delivery. It leverages the decoupled compute power and storage capability in the Microsoft Azure to support the IFRS 17 Solution outcomes.

As we define in Data warehouse design principle , the ETL Framework, for the IFRS 17 solution consist of the following five (5) layers ( Landing , Staging , Abstraction , Foundation and Insight) within a single Azure SQL DW instance, where the solution components will be developed:

### 5.2.1 Source Extraction Considerations for LBU IT

The Landing Layer of the ETL Framework is dependent on the:

* LBU-specific Source Extraction Programs will be generating the Source Data File and the corresponding Control File and;
* Specific LBU IT Teams delivering the Source Data File and the corresponding Control File to the Landing Layer of the Data Lake.

As per guiding principles, for each LBU IT Team to adhere to, in order to, support the source data requirements of the IFRS 17 Solution

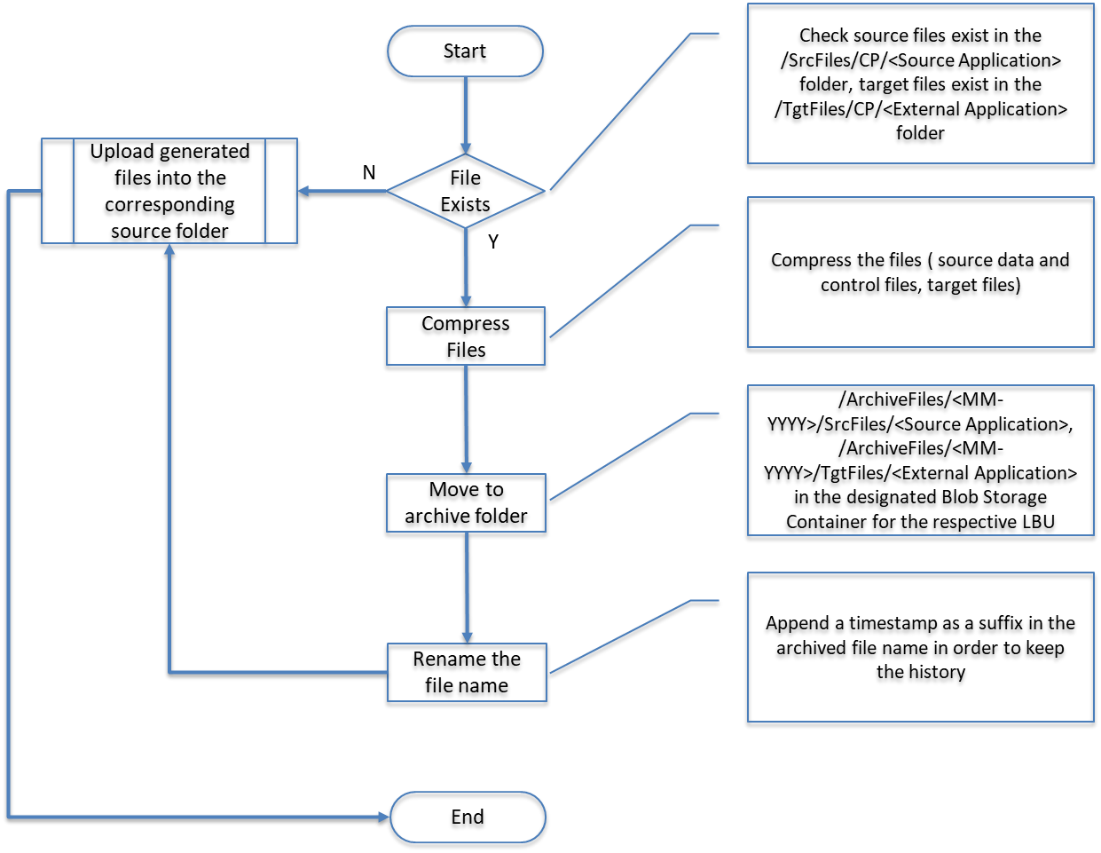
* Generate source system data into Source Data Files and a corresponding control files, both in common file format, e.g. TXT, CSV, XLS, etc.
* Each source system’s Source Extraction Program would generate one (1) or more Source Data Files per source system.
* Each data file generated by the Source Extraction program will have a corresponding Control File.
* Each LBU IT team will be landing the Source Data File and its corresponding Control File in the Landing Layer of the Azure Data Lake Gen2.
* Each LBU will have its own Landing Layer, which will be a Blob Storage Contained in the Azure Data Lake Gen2 service, for the respective LBU. The respective LBU IT Teams will be landing the files in their own Blob Storage Container.
* Each LBU IT team will use the Secure File Transfer Protocol (SFTP) to move the files from on premise to the Azure Data Lake Gen2.
* Each LBU IT team will be responsible for the:
* Extraction of the Data Source File and its corresponding Control File.
* Landing the files into the respective LBU Blob Storage Containers on Azure Data Lake Gen2 by MoveIT.
* When the Source Data File and Control File is delivered to the Landing Layer by the specific LBU IT teams, they responsible to ensure:
* File-level validation checks are done for the following:
* No duplicate data files or control files are delivered to the Source Folder in the Landing Layer. Please refer to the Folder Structure in the next section.
* Validate file-level checksums upon landing the file into the Landing Zone. Ensure data files and control file file-level integrity is not compromised (i.e. files are no corrupted) whilst the files are being transferred to the Azure Data Lake.
* Ensure the guaranteed delivery of the data file and control file to the Data Lake.
* A PowerShell script that will be developed by LBU IT team prefer to do the above file-level validation.
* Whenever, any of the above fails, the specific LBU IT teams will need to log the details and an email notification will be sent to a distribution list of the specified LBU IT team.

### 5.2.2 Landing Layer Folder Structure recommendation

The Landing Layer, which will be on the Blob Storage Container of the Azure Data Lake Gen2. It will have the following root folders with subfolders defined to support the solution:

* Source Files – SrcFiles:
* Root Source Files directory (SrcFiles) in the LBU (e.g. phk) Blob Storage Container.
* It will have sub-directories by Source Applications.
* The respective LBU Source Extraction Program will land the Source Data Files into the respective folders by Source Application.
* Each Source File will have a corresponding Control File. The respective Source Application folders in the Source Files (SrcFiles) folder is populated with the Source Data and corresponding Control Files by MoveIT.
* The steps for MoveIT to move the files generated by the Source Extraction programs:
  + The Source Extractions Programs would write the Source Data File and corresponding Control File into an on premise Fileserver.
  + If Source Data File and corresponding Control File do exist, please refer to steps under Archive File (ArchiveFiles) section below to archive existing files.
  + MoveIT will need to ensure that no Data Files and corresponding Control Files exist in each of the respective Application Folders within the Current Period (CP) in the Source Files (SrcFiles) root folder, /SrcFiles/CP/<Application Folders>.
  + MoveIT will move the data from the on premise Fileserver to the /SrcFiles/CP/<Application Folders> folder in LBU specific Blob Storage Container of the Azure Data Lake Gen2.
* Archive Files – ArchiveFiles:
* Root Archive Files directory (ArchiveFiles) in the LBU (e.g. phk) Blob Storage Container. It will have sub-directories by Source Applications, identical to the Source Files (SrcFiles) folder.
* The steps for MoveIT to move the files from the Current Period folder (CP) in the Source Files (SrcFiles) and Target Files (TgtFiles) to the Archive Accounting Period in the Archive Files (ArchiveFiles) folder:
  + If source files exist in the /SrcFiles/CP/<Source Application> folder, or target files exist in the /TgtFiles/CP/<External Application> folder, MoveIT will need to compress the files (data and control files) and;
  + Move the compressed files (data and control files) to the Archive File folder, /ArchiveFiles/<MM-YYYY>/SrcFiles/<Source Application> or /ArchiveFiles/<MM-YYYY>/TgtFiles/<External Application> in the designated Blob Storage Container for the respective LBU.
  + If multiple files are sent for the same accounting period, each compressed file name will append a timestamp as a suffix in order to keep the history always.
  + The suggested data retention period for Archive data is 7 years in the Data Lake.
* Target Files – TgtFiles:
* Root Target Files directory (TgtFiles) in the LBU (e.g. phk) Blob Storage Container.
* It will have sub-directories by Target Applications that it will be delivering data via file exchange.
* According to the archiving mechanism of Source Files, same archiving mechanism will also be applied in the Target Files.
* This is subject to refinement upon finalization of the consumption pattern of the target applications.
* Log Files – LogFiles
* Root Log Files directory (LogFiles) in the LBU (e.g. phk) Blob Storage Container.
* It will have sub-directories by ETL Applications to store log files per application
* This is subject to refinement; the suggestion is to housekeep log files on a monthly basis.

A flowchart for source data, control files and target files archive process:



Below diagram is the folder structure for one LBU in the ADL, detail information for each directory please refer to the below remarks.



The detail Folder structure can refer the Naming standard document

## 5.3 ETL Audit, Balance and Control Framework (ABC Framework)

ETL Audit, Balance and Control framework lay in their core supporting for dynamic configurations, logging, execution status tracking and restarting or recovering mechanism.

This framework stores all metadata of ETL control objects for ETL activity:

* ETL Audit, Balance and Control Metadata Tables – Consists of technical metadata tables for driving the core ETL process's execution for adapting all LBU.
* Batch data links between logic layers – A set of audit control fields representing one batch data flow. These values are stored in each tables and provides a link between the destination dataset and the process that created or modified the dataset.

*1. Batch Number*

*2. Accounting period*

*3. Execution date*

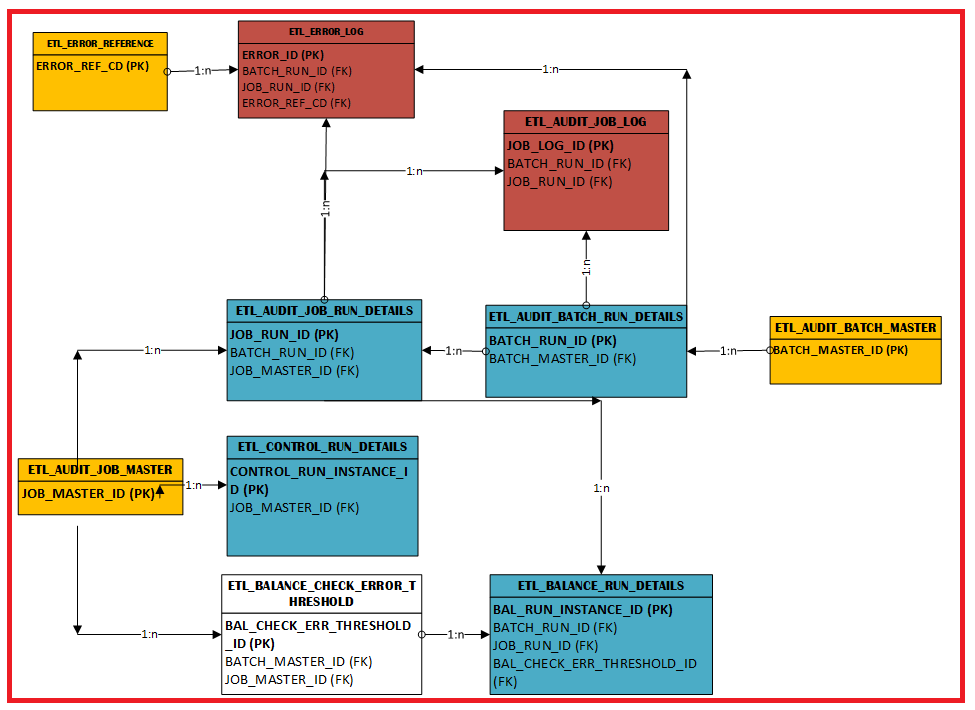
*4. Start time*

*5. End time*

*6. Version*

*7. Source system*

* Control-M orchestration – SSIS Packages will be wrapped into control-m jobs for orchestrating the whole ETL processes via Control-M. The exact method please refer to the below Control-M section.
* **All data table in Staging, Abstraction, foundation should contain BATCH\_RUN\_ID and JOB\_RUN\_ID to handle the ABC framework**



**Approach:**

Based on the ETL parameters which will be defined in the JSON configuration file in the Control-M side) and related configuration settings for one LBU, the pass in parameter value will drive the ETL 'core' packages (ETL1 ~ ETL7) running. And an ETL process (SSIS package) including pre-process and post-process will record related key running information like execution status in the metadata table for tracking purpose. ETL job orchestration (depends on the job dependencies) will be integrated and controlled by the Control-M Jobs.

The details please refer to another document "Pru\_PCA\_IFRS17\_Audit\_Balance\_Control\_Error\_Handling\_framework".

### 5.3.1 ABC Audit Process Flow



### 5.3.2 Record Count Check Flow (Landing to Staging)



### 5.3.3 Record Count Check Flow (Staging to Foundation)



**5.3.4 Control Process Flow**



**5.3.4** **Error and exception Framework**

Error handling could be divided into 2 sections Error Prevention and Error Response

**5.3.4.1 Error Prevention Flow**



**5.3.4.1 Error** **Response Flow**



# 5.4 CI/CD & Control-M Approach

SSIS integrates with many source code control software and so any integration with such kind of technology will mean much cleaner source code control and hence much automated deployments.

If source control software is installed on the computer, select add to source control to associate the project with source control.

Diagram 1: MoveIT job trigger approach

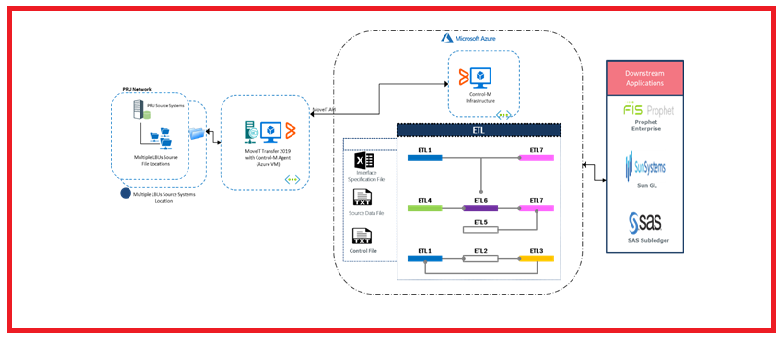
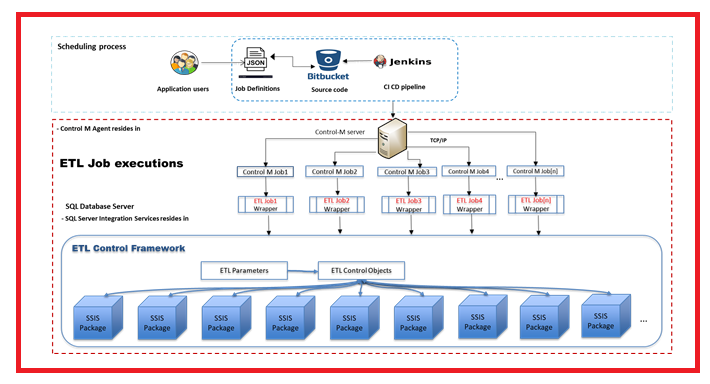


Diagram 2: ETL Jobs orchestration approach.

****

# 6.0 References

| **Document Title** | **Document ID** | **Date** | **Publishing Organization** | **Version** |
| --- | --- | --- | --- | --- |
| SDL122\_PCA\_IFRS17\_DD\_SSIS Coding Standard\_Best Practices and Naming Convention\_v1.0.docx | <ID> | < 20-12-2019 > | Prudential IFRS 17 Project | V1.0 |
|  |  |  |  |  |
|  |  |  |  |  |

1. **Document References**

# 7.0 Assumptions

* All intermediate data stores will be using Azure SQL DW
* Metadata DB objects of ETL Audit, Balance, Control and Error Handling Framework are stored into a database in a separate database instance
* SAS landing area will be a separate Azure SQL DW instance
* SAS results marts are file based
* Sun-GL database is running on SQL Server on premise edition, running Azure VMs.