SSIS Audit Balance Control & Error Handling framework

**Version 1.4**

**Document Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable Title:** | Pru\_PCA\_IFRS17\_Audit\_Balance\_Control\_Error\_Handling\_framework | | |
| **Owner:** | Sudhakar Dhulipudi | **Deliverable ID:** |  |
| **Deloitte Reviewer:** | Sudhakar Dhulipudi  Subodh Ojha  Manash Ranjan | **Review Date:** | 4/2/2020 |
| **Prudential Reviewer:** | William YP Chan  Edward SP Chan  Matthew SL Yeung  Leo MW Lau | **Review Date:** | 4/3/2020 |
| **Prudential Approver:** | Benny WK Ho | **Sign-Off Date:** | Email dated Apr 3rd 2020 02:07 PM HKT |

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Version Development** | **Author** |
| 05-Dec-2019 | 0.1 | Draft version | Neha Dosaj |
| 20-Dec-2019 | 0.2 | Initial Version | Neha Dosaj |
| 13-Feb-2020 | 0.3 | Updates based on preliminary review | Puja Narendra Rao/ Vikram Reddy |
| 27-March-2020 | 0.4 | Updates based on review with Prudential Team  Updated Balancing (3.2.2) and Control (3.2.3) Section  Modified ABC Data models in section 3.1 | Puja Narendra Rao/ Vikram Reddy |
| 03-April-2020 | 1.0 | Base line version  Update based on review with Prudential Team.  Updated Implementation Steps.  Updated Data Model in section 3.1  Updated flow balancing flow diagrams in sec 3.2.2  Updated Control Process ‘Scenarios’ in section 3.2.3 | Puja Narendra Rao/ Vikram Reddy |
| 09-April-2020 | 1.1 | Updated table and column names as per data modelling naming standards | Puja Narendra |
| 10-April-2020 | 1.2 | Embedded ABC Implementation checklist in Section 5 | Puja Narendra |
| 06-May-2020 | 1.3 | Included Error Report POC | Puja Narendra |
| 26-May-2020 | 1.4 | Updated implementation steps and checklist for new error checks | Puja Narendra |

Table of contents

[1. Introduction 1](#_Toc40969568)

[2. Assumptions and Scope 1](#_Toc40969569)

[2.1 Out of Scope Items 1](#_Toc40969570)

[3. ETL ABC and Error Handling Framework 1](#_Toc40969571)

[3.1 Metadata tables 2](#_Toc40969572)

[3.2 Audit, Balance and Control Process 3](#_Toc40969573)

[3.2.1 Audit Process 4](#_Toc40969574)

[3.2.2 Balance Process 6](#_Toc40969575)

[3.2.3 Control Process 10](#_Toc40969576)

[4. Error and exception Framework 14](#_Toc40969577)

[4.1 Error Prevention 14](#_Toc40969578)

[4.1.1 Source File Checks 15](#_Toc40969579)

[4.1.2 Checks done during the ETL Processes 15](#_Toc40969580)

[4.2 Error Response 16](#_Toc40969581)

[4.2.1 Common steps for Error Response 16](#_Toc40969582)

[4.3 Error Report 16](#_Toc40969583)

[5. ABC and Error Handling implementation with ETL4 as Example 18](#_Toc40969584)

[Step 1: Collect and consolidate SSIS Package dependency matrix for all ETLs 18](#_Toc40969585)

[Step 2: Identify Batch and Job 18](#_Toc40969586)

[Step 3: Populate Metadata in Master tables 18](#_Toc40969587)

[Step 4: Add and map BATCH\_RUN\_ID and JOB\_RUN\_ID to the tables 19](#_Toc40969588)

[Step 5: Include all the required ABC Framework packages to your project/solution. 19](#_Toc40969589)

[Step 6: Add the pre-processing and post-processing activities for each package/job. 19](#_Toc40969590)

[Step 7: Identify and Create custom balancing component, if required 20](#_Toc40969591)

[Step 8: Identify and Create custom error checks, if required 20](#_Toc40969592)

[Step 9: Assign values to package variables 20](#_Toc40969593)

[Step 10: Add Error Event Handlers. 21](#_Toc40969594)

[Step 11: Create package parameters 21](#_Toc40969595)

[Step 12: Parameterization of connection strings, schema names, etc 21](#_Toc40969596)

[6. Appendix 22](#_Toc40969597)

[6.1 Acronyms 22](#_Toc40969598)

[6.2 Performance Test results 22](#_Toc40969599)

[6.3 Additional queries: 23](#_Toc40969600)

[6.4 Sample Report: 23](#_Toc40969601)

[6.5 Issue Tracker: 24](#_Toc40969602)

[6.6 PHKL Review document: 24](#_Toc40969603)

# Introduction

This document provides technical design details for Audit Balance and Control (ABC) and Error handling framework for IFRS-17 baselining ETL#4.

# Assumptions and Scope

The scope, content, and style of this document implicitly make the following assumptions regarding applicable project environments and readers:

* This document assumes that the readers have a basic knowledge of SSIS tool and ETL concepts.
* This document assumes that the readers have some knowledge or expectations of business and technical requirements.

The scope of the document entails:

* This document comprises of design approach of ABC and error Handling framework, no code component included
* The ability to add traceability to records within data integration landscape by providing meaningful operational insights
* Provides flexible configuration, control and monitoring options
* Audit Balance and control (ABC) framework section may change as and when there is update in requirement
* Error Handling framework section may change as and when there is update in requirement

This document will be periodically reviewed and changes, for example, new processes, standards and frameworks, as and when the need arises, will be added.

## 2.1 Out of Scope Items

* Data archival
* Control-M/scheduling
* Batch orchestration
* Publishing data model
* Table Clean-up on job failure
* Functional validation and Error Checks
* ADF pipeline creation
* Job Dependencies (they would be covered as part of covered in Control-M)
* Parameterization and Passing parameters from ADF/Control-M

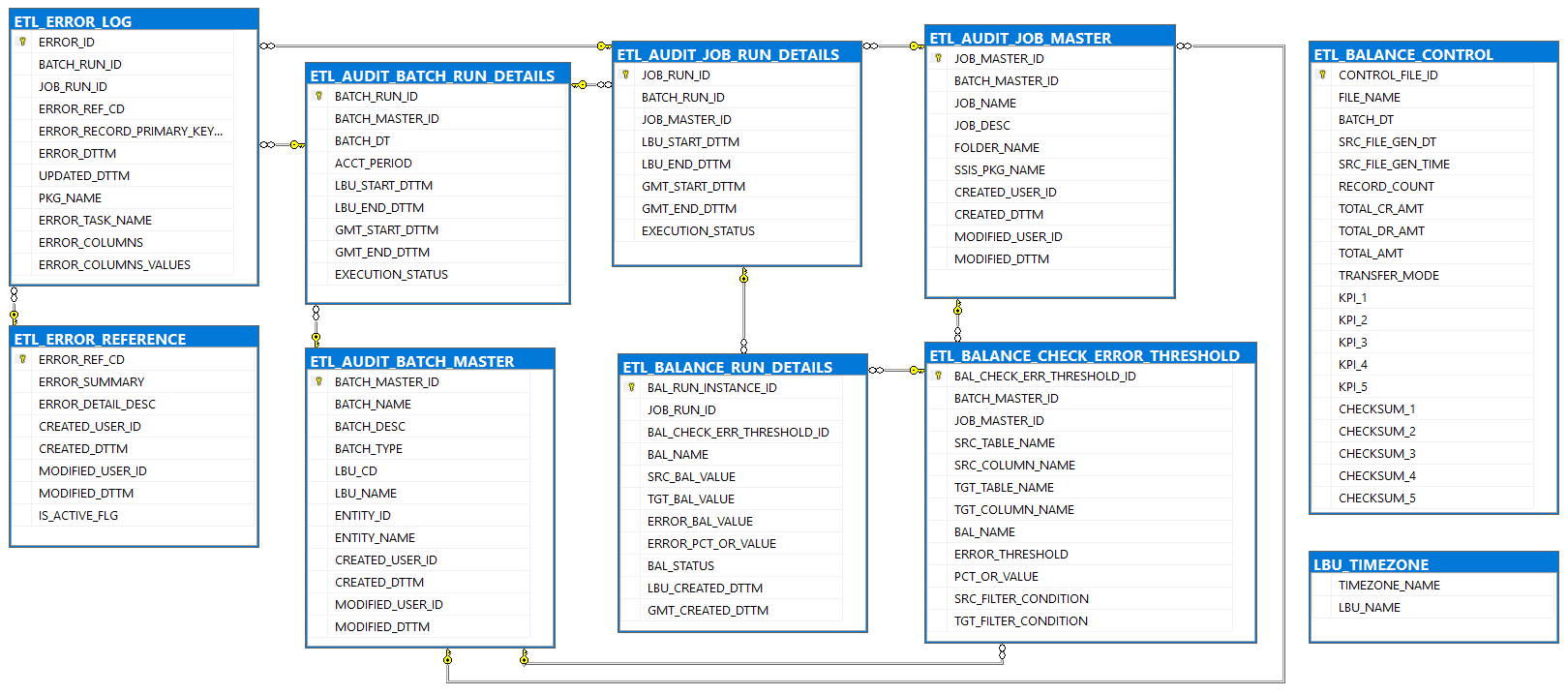
# ETL ABC and Error Handling Framework

The scope of ABC framework is to track all run statistics of ETL process. The data model is underlying set of tables to hold the metadata for framework which enables framework to perform ABC.

* The architecture comes with a metadata driven approach, leveraging Database, Scripts and SSIS capabilities, which provides flexible configuration, control and monitoring options for batch jobs and facilitates early error detection.
* Facilitate tagging of Run Time Information to target files/tables populated by the Batch Processes which adds traceability to each record loaded/updated in the files/tables.
* Reusable scripts are available to perform standard batch functionalities like batch/job execution.

## Metadata tables

Below is the data model for the ABC and Error handling framework.



|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE NAME** | **DESCRIPTION** | **SOURCE** | **OWNER** |
| ETL\_AUDIT\_BATCH\_MASTER | Batch master data | Manual entries to be made by IT team in line with Control-M process | IT |
| ETL\_AUDIT\_BATCH\_RUN\_DETAILS | Batch transactional data | Auto load by Control-M process through ADF/SSIS packages | IT |
| ETL\_AUDIT\_JOB\_MASTER | Job master data | Manual entries to be made by IT team in line with Control-M process | IT |
| ETL\_AUDIT\_JOB\_RUN\_DETAILS | Job transactional data | Auto load by Control-M process through ADF/SSIS packages | IT |
| ETL\_ERROR\_REFERENCE | Error reference data | Manual entries to be made by IT team | Business and IT |
| ETL\_ERROR\_LOG | Error transactional data | Auto load by Control-M process through ADF/SSIS packages | IT |
| ETL\_BALANCE\_CHECK\_ERROR\_THRESHOLD | Balance threshold reference data | Manual entries to be made by IT team | Business and IT |
| ETL\_BALANCE\_RUN\_DETAILS | Balance transactional data | Auto load by Control-M process through ADF/SSIS packages | IT |
| ETL\_BALANCE\_CONTROL | Interface Control File Data | Auto load by Control-M process through ADF/SSIS packages | IT |

**ETL\_AUDIT\_BATCH\_MASTER:** ETL\_AUDIT\_BATCH\_MASTER table lists various batch master data from data lake to final target system. Populating this table is typically a one-time activity.

**ETL\_AUDIT\_BATCH\_RUN\_DETAILS:** ETL\_AUDIT\_BATCH\_RUN\_DETAILS table captures all the run instances for each of the batches mentioned in the ETL\_AUDIT\_BATCH\_MASTER table.

**ETL\_AUDIT\_JOB\_MASTER:** ETL\_AUDIT\_JOB\_MASTER Table lists various job master data from data lake to final target system. Populating this table is typically a one-time activity.

**ETL\_AUDIT\_JOB\_RUN\_DETAILS**: ETL\_AUDIT\_JOB\_RUN\_DETAILS table captures all the run instances for each set of ETL steps mentioned in the ETL\_AUDIT\_JOB\_MASTER table.

**ETL\_BALANCE\_CHECK\_ERROR\_THRESHOLD:** Metadata table ETL\_BALANCE\_CHECK\_ERROR\_THRESHOLD would be used to list the error threshold values.

Note: This table would be populated based on the discussion with clients. If the percentage of records (or value) in error breaches the threshold value provided, that balance check would be considered a failure.

**ETL\_BALANCE\_RUN\_DETAILS:** ETL\_BALANCE\_RUN\_DETAILS table would be used to populate source, target and error balance values for every run. Stored procedures would check the corresponding error threshold from the BALANCE\_CHECK\_ERROR\_THRESHOLD table to determine if the check is a success or a failure.

**ETL\_ERROR\_REFERENCE:** The ETL\_ERROR\_REFERENCE table contains the detailed description of each possible data exception/error.

**ETL\_ERROR\_LOG:** Contains columns that failed validation. The values would be stored in a concatenated format along with batch/job id where the data exception/error occurred for each row.

**ETL\_BALANCE\_CONTROL:** This table is used to load the interface control file that we receive along with the data file from the source system. This file would be used to perform balance check between the data lake and staging layer.

Data Dictionary, source to target mapping and an example data set for ABC and error handling tables is attached below.



The DDLs for these tables are provided in below files:



## Audit, Balance and Control Process

Complete process flow diagram for Audit, Balance, Control and Error Handling process –



### 3.2.1 Audit Process

Control framework is responsible to invoke SSIS packages as per the schedule. When Control framework invokes an SSIS package, it generates a BATCH\_RUN\_ID using the USP\_BatchStart stored procedure and provides it to the USP\_ProcessStart stored procedure. USP\_ProcessStart stored procedure puts this BATCH\_RUN\_ID in the ETL\_AUDIT\_JOB\_RUN\_DETAILS table and generates JOB\_RUN\_ID.

USP\_BatchStart stored procedure creates an entry into the ETL\_AUDIT\_BATCH\_RUN\_DETAILS table.

It uses an incremental value BATCH\_RUN\_ID, respective value for BATCH\_MASTER\_ID from ETL\_AUDIT\_BATCH\_RUN\_DETAILS table. The BATCH\_DATE would be taken from the JSON Parameter file, ACCT\_PERIOD from the Control File, START\_DATE\_TIME\_LBU and START\_DATE\_TIME\_GMT is the current time in LBU time zone and in GMT respectively. The END\_DATE\_TIME\_LBU and END\_DATE\_TIME\_GMT is left blank. The EXECUTION\_STATUS is set to ‘RUN’.

ETL\_AUDIT\_BATCH\_MASTER-



ETL\_AUDIT\_BATCH\_RUN\_DETAILS -



USP\_ProcessStart stored procedure creates an entry into the ETL\_AUDIT\_JOB\_RUN\_DETAILS table. It uses an incremental value JOB\_RUN\_ID, respective value for BATCH\_RUN\_ ID from ETL\_AUDIT\_BATCH\_RUN\_DETAILS table, respective value for JOB\_MASTER\_ID from ETL\_AUDIT\_JOB\_MASTER table. The START\_DATE\_TIME\_LBU and START\_DATE\_TIME\_GMT is the current time in LBU time zone and in GMT respectively. The END\_DATE\_TIME\_LBU and END\_DATE\_TIME\_GMT is left blank. The EXECUTION\_STATUS is set to ‘RUN’.

ETL\_AUDIT\_JOB\_MASTER –



ETL\_AUDIT\_JOB\_RUN\_DETAILS –

.

Once the package run is complete, it would call the USP\_ProcessSuccess/USP\_ProcessFailure stored procedure followed by USP\_BatchSuccess/USP\_BatchFailure stored procedure, passing information on the run status (Success/Failure).

Stored procedure would put the END\_DATE\_TIME\_LBU and END\_DATE\_TIME\_GMT in both the tables (at the time when this record is being updated with the status).





Below is the flow for the Audit Process-



### 3.2.2 Balance Process

Balance part of ABC framework deals with mechanism to validate that all records are processed by a job/package (Record Count check). It also validates that Key Performance Indicator attribute values are captured completely (KPI Balance Check).

**Record Count check:**

**Balance check to see if** **Source record count = Target record count + Error record count(if applicable)**

This check validates that all records at source are accounted for, in target and error tables. For example, if there are 1000 source records for an ETL, 1000 records in target and 0 in error, it is a successful record count check as we have accounted for the total 1000 source records in Target and Error tables. Say, out of 1000 records, 1000 records make it to target and 50 go to error, it would result in Record count check failure.

**KPI Balance Check:**

There may be multiple KPI attributes from the source system that are loaded to target. It is critical to ensure data of these KPI attributes are completely captured and loaded to target. Any drop in data on the identified KPIs should be captured and reported.

Not all jobs may deal with KPIs, so not all jobs would have a corresponding KPI Balance check. KPI Balance check would be applied on pre-decided set of jobs and pre-decided set of KPI attributes from source to target.

KPI Balance check would do the below check:

**Source KPI Value = Target KPI Value + Error KPI value (if applicable)**

**Or**

**Error KPI Value < Threshold value/percentage**

**Threshold value/percentage** here is used to account for any rounding error.

**Date Lake to Staging layer:**

**Balance on data files when a separate “control” file is provided along with the data file:**

This control file will have Record count and Source KPI Values mentioned in it, in a predefined structure. Accordingly, the kind of balance to do – (Record count check, KPI balance check), the column and table values would be passed as parameters from the ETL to the framework. So, any addition/deletion/update of Balance checks would need corresponding change at ETL as well.

Both the below mentioned checks would be applied on the data file using the “control” data from the control file. This way, ETL doesn’t need to determine the source record count and Source KPI Value using the data file.

**Source record count = Target record count + Error record count (And the check on whether the error record count or percentage breaches threshold)**

**Source KPI Value = Target KPI Value + Error KPI Value (Plus or Minus the sum threshold value/percentage)**

**Sample scenario for ETL4:**

KPI attributes and threshold values are inserted into ETL\_BALANCE\_CHECK\_ERROR\_THRESHOLD table as shown below –

ETL\_BALANCE\_CHECK\_ERROR\_THRESHOLD -



According to the data entered in the ETL\_BALANCE\_CHECK\_ERROR\_THRESHOLD table, the threshold percentage for record count is 0% and for KPI attributes is 10% i.e., the balance is reported as a ‘FAILURE’ whenever there is a count mismatch or when the Error KPI Value is greater than 10%.

Data from control files would be loaded to a control table in Audit area(AUDIT\_<xxxxx> schema). The structure of control table would be similar to the structure of the control file with a few extra buffer columns.

ETL\_BALANCE\_CONTROL –



Once control information is present in the Audit area along with the data from files in Staging area, Balance checks (Record Count check and KPI Balance check) would be applied as explained in above sections.



As shown above, for JOB\_RUN\_ID = 1 which is the Landing to Staging run, the record count and KPI sum are equal in the source and target table. Hence, the 3 records have BAL\_RESULT = ‘SUCCESS’.

For JOB\_RUN\_ID = 6, the difference between the number of records in source (Staging) and target (Abstraction) is greater than 0 because of which the BAL\_RESULT = ‘FAILURE’. However, the difference in KPI sum is 2.29% which is lesser than the threshold percentage of 10% because of which the BAL\_RESULT = ‘SUCCESS’.

**The job should only be allowed to continue with permitted threshold when there is no anticipated data integrity or data dependencies within the batch data set.**

**The job and respective batch will be forced to fail upon the threshold limit breach.**

This flow for all layers (Landing to Staging (Control File available), Staging to Abstraction (Control File unavailable) and Abstraction to Foundation (Control File unavailable) is explained below:

**Data Lake to Staging (Control File available) -**

**Record Count Check -**



**KPI Balance Check -**



**Table to table (Control File unavailable) -**

**Record Count Check -**



**KPI Balance Check -**



### 3.2.3 Control Process

The major control process for ETL will be taken care from scheduling orchestrator (control M) . The ETL process orchestration and scheduling tool, will schedule all SSIS packages of ETL1 ~ ETL7 based on the execution dependencies. Between the Control-M job and the SSIS objects (ADF), wrapper objects (scripts & control table) as a “bridge” to connect them.

* Application users will check-in standard template in JSON format to run job as code.
* JSON file template will be validated by scheduling team before triggering Jenkin job.
* CICD(Jenkin) pipeline will trigger the jobs to be deployed in Control-M server.
* Control-M Jobs will expose to Azure Data Factory via wrapper script (power-shell script) and perform operation.

**Sample:** Invoke-AzDataFactoryV2Pipeline -ResourceGroupName "rsg-sgrass-dev-az1-9apb4c" -DataFactoryName "dtf-sgrass-dev-az1-9apb4c-1" -PipelineName "Schema\_Parameterization\_Demo" -ParameterFile .\ParamTest.json

The **Invoke-AzDataFactoryV2Pipeline** command starts a run on the specified pipeline and returns a ID for that run. This GUID can be passed to **Get-AzDataFactoryV2PipelineRun** or **Get-AzDataFactoryV2ActivityRun** to obtain further details about this run.

PS C:\> Get-AzDataFactoryV2PipelineRun -ResourceGroupName "ADF" -DataFactoryName "WikiADF" -PipelineRunId "61eb095a-fe23-4591-8a97-fade6c65ca72"

ResourceGroupName : ADF

DataFactoryName : WikiADF

RunId : 61eb095a-fe23-4591-8a97-fade6c65ca72

PipelineName : DPWikisample

LastUpdated : 9/14/2017 12:21:02 AM

Parameters : {[url, http://adfsamplewebapi.azurewebsites.net/api/execute/sample]}

RunStart : 9/14/2017 12:20:54 AM

RunEnd : 9/14/2017 12:21:02 AM

DurationInMs : 8246

Status : Succeeded

Message :

PS C:\> Get-AzDataFactoryV2ActivityRun -ResourceGroupName "ADF" -DataFactoryName "WikiADF" -PipelineRunId "f288712d-fb08-4cb8-96ef-82d3b9b30621" -RunStartedAfter "2017-09-01" -RunStartedBefore "2017-09-30"

ResourceGroupName : ADF

DataFactoryName : WikiADF

ActivityName : MyWebActivity

PipelineRunId : f288712d-fb08-4cb8-96ef-82d3b9b30621

PipelineName : DPWikisample

Input : {method, url, headers, body...}

Output : {operationstatus}

ActivityRunStart : 9/14/2017 12:20:57 AM

ActivityRunEnd : 9/14/2017 12:21:00 AM

DurationInMs : 2768

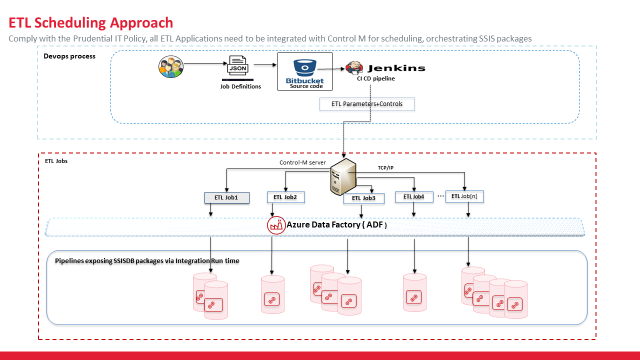
Status : Succeeded

Error : {errorCode, message, failureType, target}

* Input parameters will be supplied from JSON (meta-data) to CTM Jobs as in-condition.

**Sample:**

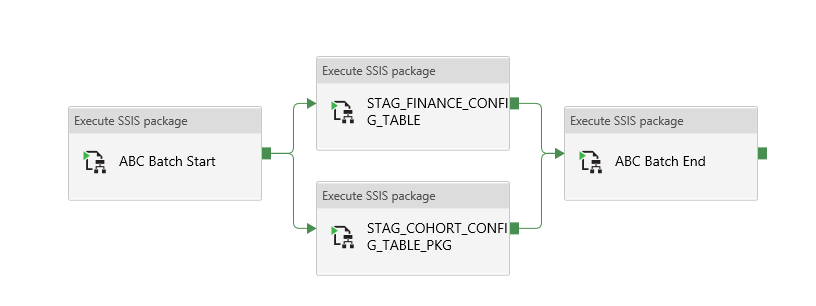




Control-M deals with Batch runs, Restartability, manually stopping/starting a job, handling how jobs would run in sequence and in parallel etc.

Scheduling-to decide whether the batch should run on a day and if yes (Example: Batch might not need to run on US holidays), at what time.

The below diagram depicts the ADF implementation of ABC packages. The ‘ABC Batch Start’ and ‘ABC Batch End’ tasks call the LOGG\_BATCH\_START and LOGG\_BATCH\_END packages respectively



The control process flow is explained below.



Here are the various scenarios and how control process works in these scenarios.

**Scenario 1: Fresh Run:**

When a Batch is started an entry in made into the ETL\_AUDIT\_BATCH\_RUN\_DETAILS table with a new BATCH\_RUN\_ID. When a package is triggered, the control task would pass the control to the following tasks as there would be no entries in the ETL\_AUDIT\_JOB\_RUN\_DETAILS table for that job with the current BATCH\_RUN\_ID.

**Scenario 2: Job has succeeded, but the batch has failed due to another job in the batch:**

When a job completes successfully without any error and passes the balance checks, the JOB\_RUN\_ID generated for that job is set to ‘SUCCESS’ in the ETL\_AUDIT\_JOB\_RUN\_DETAILS table. In this case, if the batch is restarted, the framework prevents the succeeded job from re-running as it would have an entry in the ETL\_AUDIT\_JOB\_RUN\_DETAILS with the status as ‘SUCCESSFUL’ for the current BATCH\_RUN\_ID.

**Scenario 3: Failure scenario:**

In case of failure, EXECUTION\_STATUS column in the ETL\_AUDIT\_JOB\_RUN\_DETAILS would be set to “FAILED” for the failed job. When the error is fixed and the batch is re-run, the framework will skip the jobs that ran successfully and start from sailed job and proceed further.

In case for any specific reason, we need to rerun the Successfully ran Jobs also then we need to,

a. Cleanup the data loaded by the successfully ran Jobs

b. Update the Job run detail table with Status as Failed for all the applicable Jobs

c. Rerun the Batch

**Scenario 4: Full Batch re-run:**

Say a decision is taken to do a full batch re-run because of serious data issues across all source systems – even though the run status is Success on all jobs in that Batch. In this case, a new BATCH\_RUN\_ID would be created by the Audit framework, this would be treated as a fresh run scenario as we have generated a new BATCH\_RUN\_ID. A manual cleanup of data may be required before the Batch rerun to avoid duplicate records.

1. New BATCH\_RUN\_ID will not be created for the same BATCH\_DATE in case of rerun due to failure (Scenario 3)

Note:

1. Any rerun would be triggered through Control M only
2. Batch rerun will have same Batch date since same parameter file is expected to use.
3. Rerun is allowed at Job/Sub package level, Job level rerun scenario explained in the scenario 3.

# Error and exception Framework

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

## Error Prevention

Measures that stop general errors from occurring or bypass the portion when it detects an error occurs



### 4.1.1 Source File Checks

Below are the checks that should be implemented for when the source is a file

**1. File Format Checks** - Check if the number of columns in each row of the file match the required number of columns.

**2. File naming Conventions** - Structure should follow the expected convention. Incorrect format can result in processes failures or incorrect data loads.

**3. Data format Issues**- Incorrect date format or other data types can cause failures while reading the file. These issues can be addressed by doing a pre-scan of the document before starting the ETL process

### 4.1.2 Checks done during the ETL Processes

Below are the checks/actions that should be performed as part of this process

1. **Minimum columns check**-Checking if the minimum required columns are present in the source data can prevent errors in downstream processes such as calculations for mandatory column (in target tables) or primary key generation
2. **Null/Blank value handling**- Set appropriate default values such as ‘UNK’, ‘NA’ or ‘0’ to columns that have missing or null values
3. **Data format check**- Check if columns are truncated or not in expected format. This should be done for columns that would be used as primary keys, foreign keys or in join conditions.
4. **Business Rule Validation** -Certain columns, defined by the business, should not contain null/blank values or should be in the format defined by the business. Note this would require custom components based on requirements and would not be part of the reusable ABC framework.

If any of the records do not pass the above validations, they should be moved to the error tables defined in the next section.

## Error Response

Measure that should be taken when an error occurs such as Email notification or updating Audit table

### 4.2.1 Common steps for Error Response

1. **Triggers to start clean-up process** - For any SSIS package that fails, an appropriate clean-up process would need to be run based each package/ETL need. Note that there would **not** be any reusable ABC framework for the clean up the target tables

2. **Audit table entry** - An entry should be made in the Batch and Job Audit tables indicating that the Batch/Job has encountered a failure.

3. **Notification** - Optional process depends on cases to case base for each package/ETL. (This is also not part of the reusable ABC framework. Custom components would need to be developed based on each specific requirements)

## Error Report

The data in ETL\_ERROR\_LOG table could be used for generating error reports.

Here are a few sample error reports that have been generated for multiple scenarios.

These reports are generated on Staging.STAG\_DTMTM\_BENEFIT\_GL\_TEMP table in which the mandatory fields are SUN\_CD, AMOUNT\_CCY and REASON.

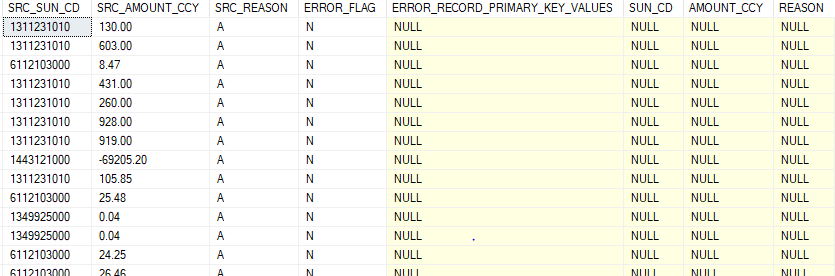
**Scenario 1:** No errors detected in the table i.e., ETL\_ERROR\_LOG table is empty for the current JOB\_RUN\_ID.

Source mandatory fields

Error Flag

Checked columns / Error Flags

Primary Key in Source table



In this scenario, all the checked columns are NULL because all the mandatory fields in the source table have non-null values.

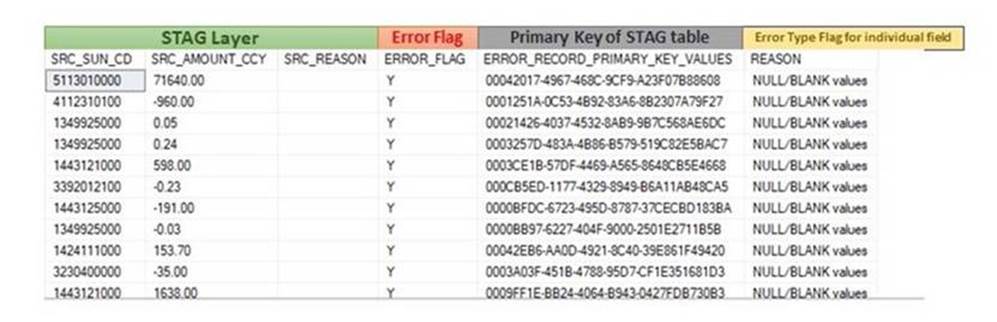
**Scenario 2:** One of the fields among the mandatory fields has a NULL/Blank value.

Checked column / Error Flag

Primary Key in Source table

Error Flag

Source mandatory fields



In this scenario, only the REASON field is blank for all records. Hence, there is only 1 Error Type Flag that is included in the report with short description as “NULL/BLANK values”.

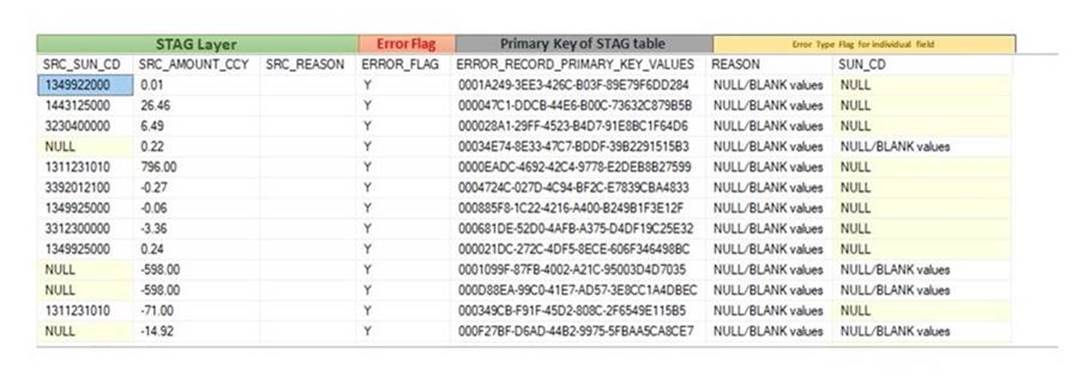
**Scenario 3:** Multiple fields among the mandatory fields have a NULL/Blank value.

Checked column / Error Flag

Source mandatory fields

Error Flag

Primary Key in Source table



In this scenario, the REASON field is blank for all records and the SUN\_CD field is null for some of the records. Hence, there are 2 Error Type Flags that are included in the report with short description as “NULL/BLANK values”.

The number of checked columns or error flags are dynamically generated in the report based on the number of columns that have errors.

The dynamic query for generating this error report is attached below –



# ABC and Error Handling implementation with ETL4 as Example

Below are the steps to implement ABC framework.

## Step 1: Collect and consolidate SSIS Package dependency matrix for all ETLs

Collect SSIS package details including dependency matrix for all ETLs and consolidate them in the attached template.



## Step 2: Identify Batch and Job

Group logically related jobs/components that always need to run together in different sets. This group of jobs would form a batch.

For Example, in ETL4 we have grouped the jobs that load Actual Cash Flow data from Data lake to Insights layer as one batch. We have grouped the jobs as Data Lake to Staging layer, Staging to Abstraction layer and Abstraction to Foundation layer.

## Step 3: Populate Metadata in Master tables

Step 3a. Create insert scripts to insert batch and job data identified in the above step into the ETL\_AUDIT\_BATCH\_MASTER and ETL\_AUDIT\_JOB\_MASTER tables.

Below is an example of entries created for ETL4 batches and jobs in the ETL\_AUDIT\_BATCH\_MASTER and ETL\_AUDIT\_JOB\_MASTER tables





Step 3b. Create insert scripts to insert the source table name, target table name, type of balance (Record Count or KPI Sum), source and target KPI column names and threshold value/percentage in the ETL\_BALANCE\_CHECK\_THRESHOLD table. As mentioned in the earlier section, this data needs to be provided by the business. If the percentage of records (or value) in error breaches the threshold value provided, that balance check would be considered a failure.

Below is an example of entries created for ETL4 in the ETL\_BALANCE\_CHECK\_THRESHOLD table for Staging and Abstraction layers.



Step 3c. Create insert scripts to insert the types of error checks to be performed.

Below is an example of entries created for ETL4 in the ETL\_ERROR\_REFERENCE table. The different types of error checks reported are job execution failure, incorrect number of columns in source file and null/blank data provided in mandatory columns.

Apart from the above validations, ETL specific custom error checks can be included in this table which might also require SSIS package modifications.



The following attachment has the template to auto-generate Insert scripts for all the Master tables.



## Step 4: Add and map BATCH\_RUN\_ID and JOB\_RUN\_ID to the tables

Add the BATCH\_RUN\_ID and JOB\_RUN\_ID fields to the tables where audit needs to be performed. Map these fields in the individual ETLs from the user-defined variables @[User::BatchRunId] and @[User::JobRunId] respectively.

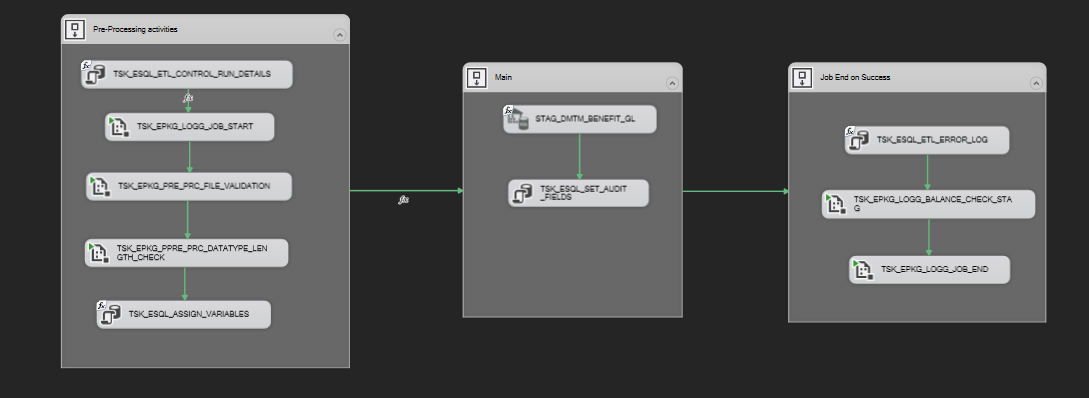
## Step 5: Include all the required ABC Framework packages to your project/solution.

Copy all the required packages from the ABC framework project to your project. Verify if all the package parameters are copied as well.

## Step 6: Add the pre-processing and post-processing activities for each package/job.

Make a copy of the job template provided and insert the code for into the main section of the template.

Below is an example implementation for ETL4 file to staging load process –



9

7

88

6

5

1

2

3

4

Please note that this is a template and all the tasks might not be required for certain jobs. For example, in the above ETL 4 implementation tasks 1, 2, 5 and 9 are mandatory. The rest of the tasks are specific for ETL4 landing to staging. Task 3, 4 and 8 are specific for file loads, hence would not be required for table loads. These are also example implementations for ETL4 and may need to be modified/customized for other ETL/jobs.

## Step 7: Identify and Create custom balancing component, if required

If additional balance checks need to be included, add them in the corresponding Balance Stored Procedures and package.

Stored Procedures for Balance check:

* USP\_BalanceCheck\_Stag – SP for calling the USP\_KPISumCheck\_Stag SP
* USP\_BalanceCheck – SP for calling the USP\_KPISumCheck\_Stag and USP\_KPISumCheck SPs
* USP\_CountCheck\_Stag – SP for count validation when Control File is available
* USP\_CountCheck – SP for count validation when Control File is unavailable
* USP\_KPISumCheck\_Stag – SP for KPI Sum validation when Control File is available
* USP\_KPISumCheck – SP for KPI Sum validation when Control File is unavailable

Packages for Balance check:

* LOGG\_BALANCE\_CHECK\_STAG.dtsx
* LOGG\_BALANCE\_CHECK.dtsx

## Step 8: Identify and Create custom error checks, if required

If additional error checks need to be included, add them in the corresponding Error Stored Procedures and package.

Stored Procedures for Error check:

* USP\_ErrorLogInsert

Packages for Error check:

* Event handler of each Staging, Abstraction and Foundation layer package
* PRE\_PRC\_FILE\_VALIDATION.dtsx
* PRE\_PRC\_DATATYPE\_LENGTH\_CHECK.dtsx

## Step 9: Assign values to package variables

The below 3 variables would be needed for all packages with pre and post processing activities

* BatchRunId (Int64)
* JobRunId (Int64)
* ControlRowCount (Int32)

If Task 3 is being used in the pre-processing task, assign values to the below package variables:

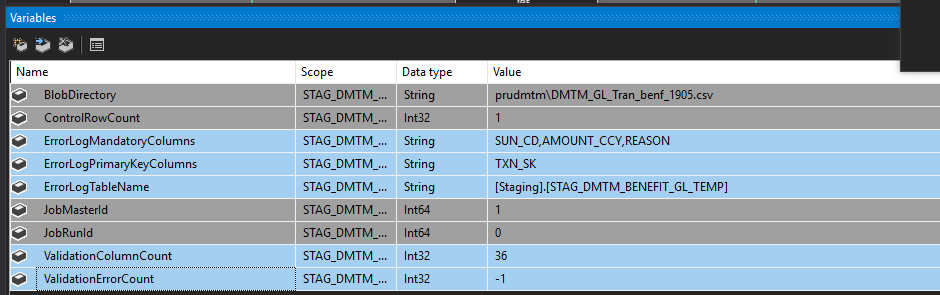
* ValidationColumnCount: This variable should have the number of columns expected in the source file.
* ValidationErrorCount: This variable will contain the number of source file validation errors. This needs to be assigned with a default value of -1.

If Task 4 is being used in the pre-processing task, assign values to the below package variables:

* DatatypeErrorCount: This variable will contain the number of datatype/length check errors. This needs to be assigned with a default value of -1.

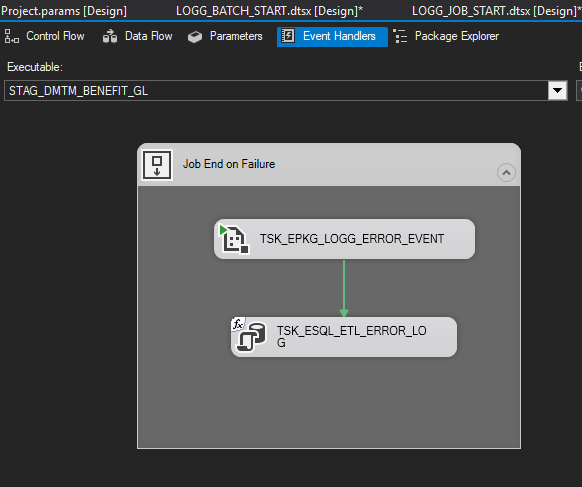
If task 7 is being used in the post-processing task, assign values to the below package variables:

* ErrorLogMandatoryColumns: This variable should have the list of columns that are mandatory in the Staging table. The column names need to be comma separated.
* ErrorLogPrimaryKeyColumns: This variable should have the names of the columns that are the primary keys in the Staging table.
* ErrorLogTableName: This variable should have the name of the target table.



## Step 10: Add Error Event Handlers.

Create an error event handler for the package if it does not already have one and add the 2 tasks (from the template package)



## Step 11: Create package parameters

Add package parameters in the LOGG\_BATCH\_START, LOGG\_BATCH\_END and the individual Staging/Abstraction/Foundation packages - “BatchName” that would hold the name of the batch from ETL\_AUDIT\_BATCH\_MASTER table and “BatchDate” that would hold the audit value from the interface control file.

## Step 12: Parameterization of connection strings, schema names, etc

Create and use the required parameters for connection strings, Schema Names etc. Following is a list of potential parameters with naming convention that may be used for the IFRS17 program. Both Source and Target can use same Schema parameter.

|  |  |
| --- | --- |
| **Parameter Type** | **Naming Convention** |
| Schema Name | param\_SCHEMA\_STAG |
| Schema Name | param\_SCHEMA\_ABST |
| Schema Name | param\_SCHEMA\_FOND |
| Schema Name | param\_SCHEMA\_FCORE |
| Schema Name | param\_SCHEMA\_INST |
| Ole DB Connection | param\_CM\_OLEDB\_ |
| ADO.NET | param\_CM\_ADONET\_ |
| Flat File | param\_CM\_FF\_ |
| Azure DW | param\_CM\_AZUREDW\_ |
| Azure Storage | param\_CM\_AZUREBLOB\_ |
| AccountPeriod | param\_ACCT\_PERIOD |
| BatchDate | param\_BATCH\_DATE |

Please refer section 9 and 10 of the below coding standards documents for additional information on using parameters.

PFB the Confluence link:

<https://collaborate.pruconnect.net/display/IFRS17Common/DevSecOps?preview=/37308090/48597514/SDL122_PCA_IFRS17_DD_SSIS%20Coding%20Standard_Best%20Practices%20and%20Naming%20Convention_v1.5.docx>

Note: The following Implementation Checklist needs to be updated by all developers during implementation of ABC Framework to individual ETLs.



# Appendix

## Acronyms

| Descriptor | Standard Abbreviation |
| --- | --- |
| SSIS | SQL Server Integration Services |
| ABC | Audit, Balance & Control |
| ETL | Extraction Transformation & Load |

## Performance Test results

Below is a quick summary of the Performance test conducted to capture the SSIS package execution time, before and after adding Batch Run id, Job Run id columns in to the target table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5M Records | | Run Time With New Columns | | | | | | Run Time Without New Columns | | | | | |
| **Step** | | **Run 1** | | **Run 2** | | **Average** | | **Run 1** | | **Run 2** | | **Average** | |
| **Pre-Processing** | | 00:00:13.0 | | 00:00:14.3 | | 00:00:13.6 | | 00:00:14.2 | | 00:00:12.3 | | 00:00:13.3 | |
| **Main** | | 00:15:36.0 | | 00:13:23.0 | | 00:14:29.5 | | 00:17:22.3 | | 00:12:34.8 | | 00:14:58.5 | |
| **Post-Processing** | | 00:00:47.8 | | 00:00:42.0 | | 00:00:44.9 | | 00:00:54.5 | | 00:00:46.6 | | 00:00:50.5 | |
| **Full Package** | | 00:16:36.8 | | 00:14:20.0 | | 00:15:28.4 | | 00:18:31.1 | | 00:13:32.8 | | 00:16:01.9 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10M Records | | Run Time With New Columns | | | | | | Run Time Without New Columns | | | | | |
| **Step** | | **Run 1** | | **Run 2** | | **Average** | | **Run 1** | | **Run 2** | | **Average** | |
| **Pre-Processing** | | 00:00:12.0 | | 00:00:10.2 | | 00:00:11.1 | | 00:00:11.5 | | 00:00:12.1 | | 00:00:11.8 | |
| **Main** | | 00:26:13.8 | | 00:25:26.8 | | 00:25:50.3 | | 00:25:32.3 | | 00:24:56.2 | | 00:25:14.2 | |
| **Post-Processing** | | 00:00:44.6 | | 00:00:40.9 | | 00:00:42.7 | | 00:00:41.3 | | 00:00:41.0 | | 00:00:41.1 | |
| **Full Package** | | 00:27:10.4 | | 00:26:18.0 | | 00:26:44.2 | | 00:26:25.1 | | 00:25:49.4 | | 00:26:07.3 | |



## Additional queries:



## Sample Report:



## Issue Tracker:



## PHKL Review document:

