Data Engineering Architecture Documentation

# Asante Financial Services Group

2023-08-31

Henry Mburu – Data Engineer

# Abstract

This document is aimed at explaining the various aspects of the data engineering function in Asante Financial Services Group.   
The document seeks to transfer knowledge regarding the data warehouse design and data pipeline processes for the purpose of understanding, reviewing or improving the existing architecture.

# Table of Contents

[Asante Financial Services Group 1](#_Toc408058102)

[Abstract 1](#_Toc1393054957)

[Table of Contents 2](#_Toc1671241264)

[CHAPTER 1: DATA WAREHOUSE 2](#_Toc1445134444)

[Overview 3](#_Toc1747630264)

[Lattice of cuboids 3](#_Toc593507467)

[Digital Credit Provider Submission Files 5](#_Toc131331242)

[Product: Safaricom Bloom 5](#_Toc408623449)

[Product: Pronto 6](#_Toc819817036)

[CHAPTER 2: DATA PIPELINES 7](#_Toc1307247181)

[Connections and Variables 7](#_Toc1920037712)

[Product: Safaricom Bloom 8](#_Toc177758322)

[DAG ID: ETL\_safaricom\_data\_dumps 8](#_Toc1713347577)

[Overview 8](#_Toc724479719)

[Configuration Parameters. 10](#_Toc2057457463)

[Tasks 10](#_Toc223237820)

[Data 16](#_Toc984145811)

[DAG ID: Bloom\_IPRS\_check 17](#_Toc1093084534)

[Overview 18](#_Toc1943168521)

[Configuration Parameters. 19](#_Toc1260311580)

[Tasks 19](#_Toc1703765242)

[Data 20](#_Toc671589087)

[DAG ID: TransUnion\_CRB\_checks 21](#_Toc2031926185)

[Overview 21](#_Toc662222518)

[Configuration Parameters. 23](#_Toc2024682957)

[Tasks 24](#_Toc709381981)

[DAG ID: ETL\_bloom\_retail\_field\_metrics 25](#_Toc1964905917)

[Overview 26](#_Toc1294371937)

[Configuration Parameters. 26](#_Toc1089557291)

[Tasks 26](#_Toc796810635)

[DAG ID: ETL\_TEP\_data 26](#_Toc1468106139)

[Overview 27](#_Toc630082825)

[Configuration Parameters. 28](#_Toc857142715)

[Tasks 28](#_Toc1602049860)

[Data 30](#_Toc1517348868)

[DAG ID: Bloom\_limit\_refresh\_push 31](#_Toc1085235893)

[Overview 32](#_Toc497568458)

[Configuration Parameters. 33](#_Toc757248063)

[Tasks 33](#_Toc2023463098)

[Data 37](#_Toc114363026)

[DAG ID: EL\_live\_bloom\_whitelist 38](#_Toc878952319)

[Overview 39](#_Toc403666213)

[Configuration Parameters. 40](#_Toc1747891335)

[Tasks 40](#_Toc847552253)

[Data 41](#_Toc1050497755)

[DAG ID: ELT\_bloom\_till\_activity\_raw\_transactions 42](#_Toc827086614)

[Overview 43](#_Toc1413181906)

[Configuration Parameters. 43](#_Toc6349637)

[Tasks 43](#_Toc1353733588)

[DAG ID: data\_dumps\_bloom\_data\_recon 43](#_Toc1859340923)

[Overview 44](#_Toc1122055613)

[Configuration Parameters. 44](#_Toc2013408528)

[Tasks 44](#_Toc478775532)

[DAG ID: mpesa\_statements\_bloom\_data\_recon 45](#_Toc1997081998)

[Overview 46](#_Toc2000209402)

[Configuration Parameters. 46](#_Toc1796674210)

[Tasks 46](#_Toc120353605)

[DAG ID: tep\_data\_bloom\_recon 47](#_Toc1011287192)

[Overview 48](#_Toc2131478222)

[Configuration Parameters. 48](#_Toc557568123)

[Tasks 48](#_Toc1851422199)

[DAG ID: ETL\_safaricom\_recons\_mifos\_posting 49](#_Toc993775360)

[Overview 49](#_Toc1334481405)

[Configuration Parameters. 50](#_Toc542783844)

[Tasks 50](#_Toc998794524)

[Product: Interswitch Uganda 51](#_Toc1248889648)

[DAG ID: scoring\_pipeline\_interswitch\_uganda 51](#_Toc1672206019)

[Overview 52](#_Toc411395938)

[Configuration Parameters. 52](#_Toc1552743789)

[Tasks 52](#_Toc91275031)

[Product: Remita 55](#_Toc921297822)

[DAG ID: scoring\_pipeline\_remita 55](#_Toc408138351)

[Overview 55](#_Toc1096142546)

[Configuration Parameters. 55](#_Toc173162286)

[Tasks 56](#_Toc877539643)

[Product: Pronto 57](#_Toc580640682)

[DAG ID: ETL\_pronto\_scoring\_results 57](#_Toc612650214)

[Configuration Parameters. 57](#_Toc1960323876)

[DAG ID: Reports 57](#_Toc1177555470)

[Tasks 58](#_Toc714707563)

[DAG ID: DVC\_investor\_reports 61](#_Toc1946893967)

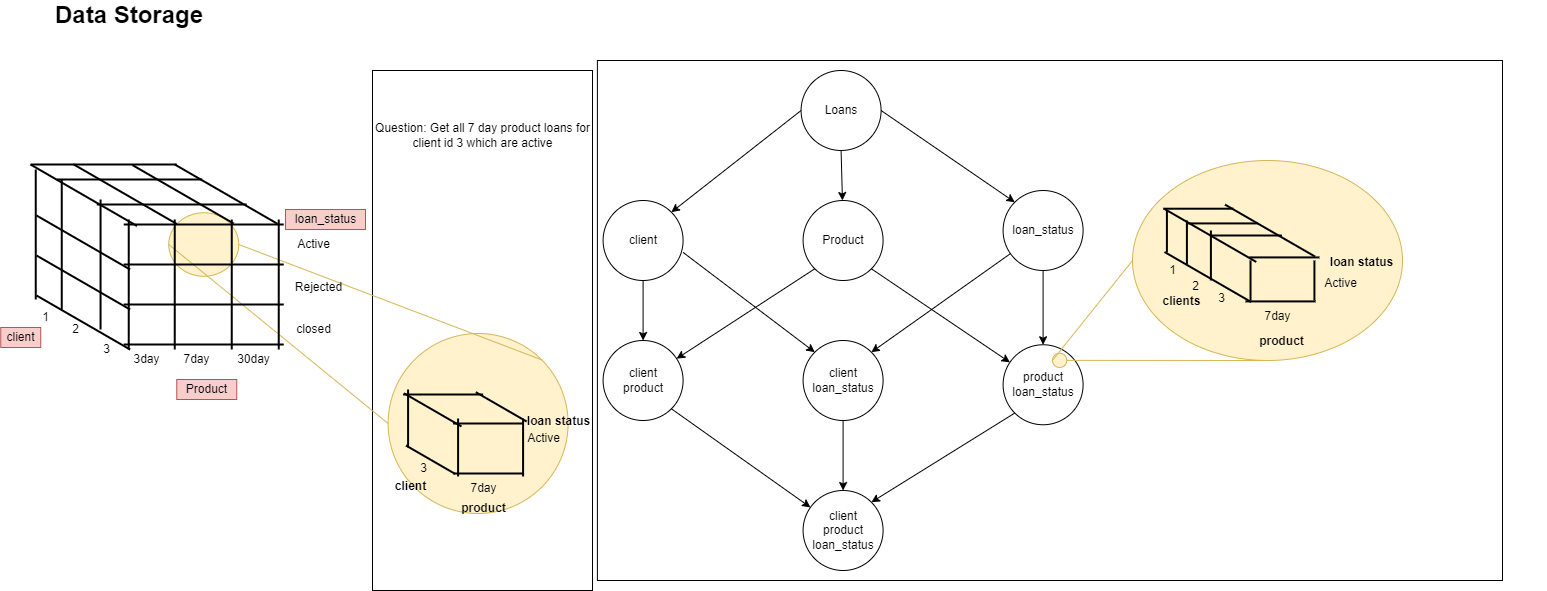
[Tasks 61](#_Toc118412150)

# CHAPTER 1: DATA WAREHOUSE

## Overview

The data warehouse at Asante FSG is designed on a [star schema topology](https://www.geeksforgeeks.org/star-schema-in-data-warehouse-modeling/). There is a central fact table in each product schema, surrounded by multiple dimension tables.

Below diagram depicts the querying logic cuboids based on the three main dimensions across all products. It shows the different combinations of data possible using the dimension keys from the fact table

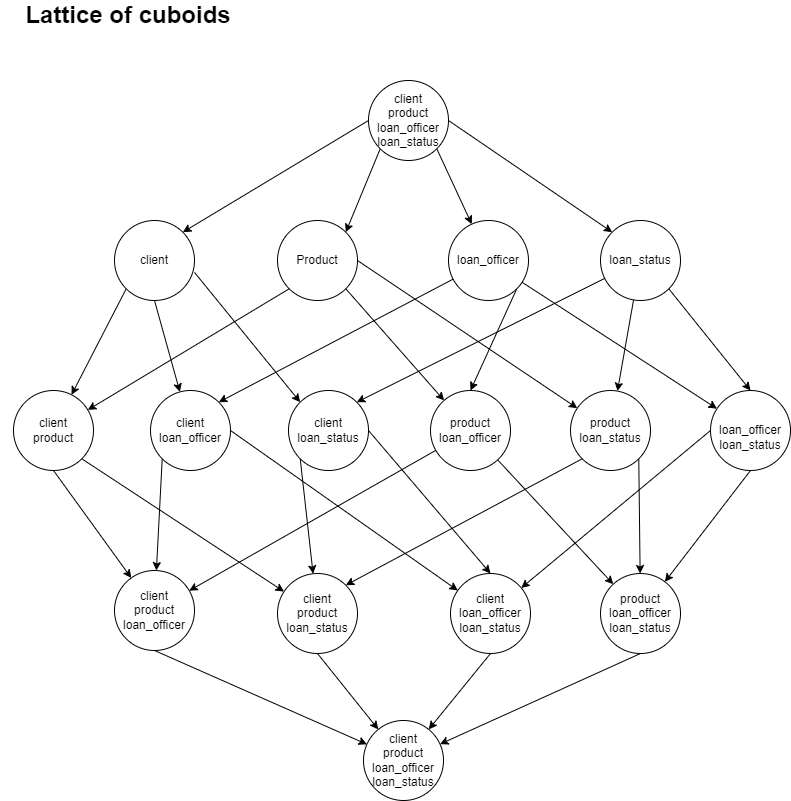


## Lattice of cuboids

Below diagram depicts the lattice of cuboids for the four main dimensions in the data warehouse. The loan officer dimension is **not always** linked to loan data but when linked, the diagram shows the possible combinations of the data for efficient slicing and dicing of the data. The loan officer dimension is the other dimension in the main dimensions

For any analysis on the loans fact table, the logic will be based on a combination or a single reference to the main dimensions, I.e, looking at a loan from the fact table, you will always be analysing the loan based on the loan status, product, loan officer, client or a combination of these perspectives.

The lattice of cuboids shows you all possible combinations based on these dimension tables.



## Digital Credit Provider Submission Files

The Data Science team is expected to provide the disbursements data, customer data, loan repayments data and non-performing loans dataset for submission to TransUnion DCP.

To retrieve the data from the data warehouse below views are used.

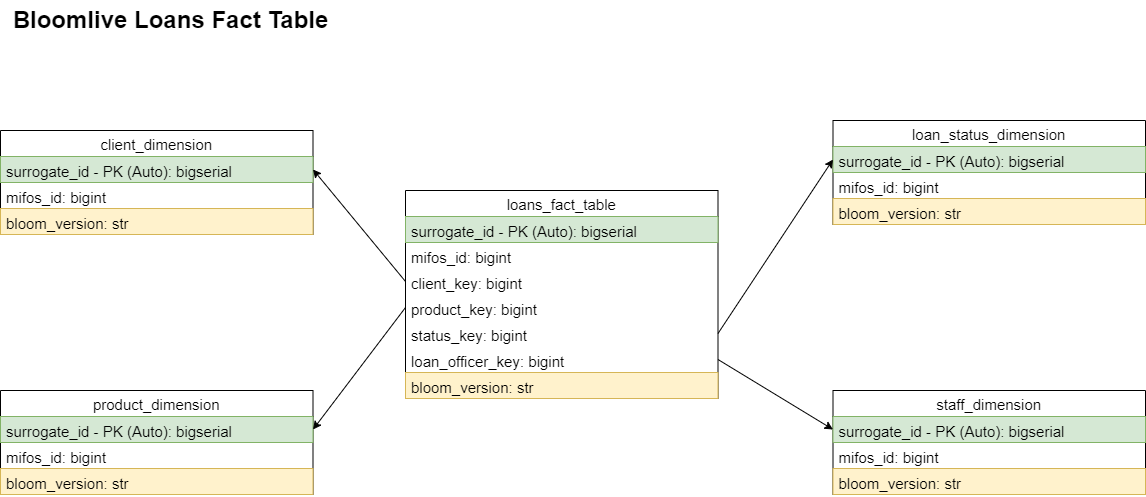
1. cbk\_customer\_data
2. cbk\_digital\_loan\_repayments\_data
3. cbk\_digital\_loans\_account\_data
4. cbk\_non\_performing\_digital\_loans

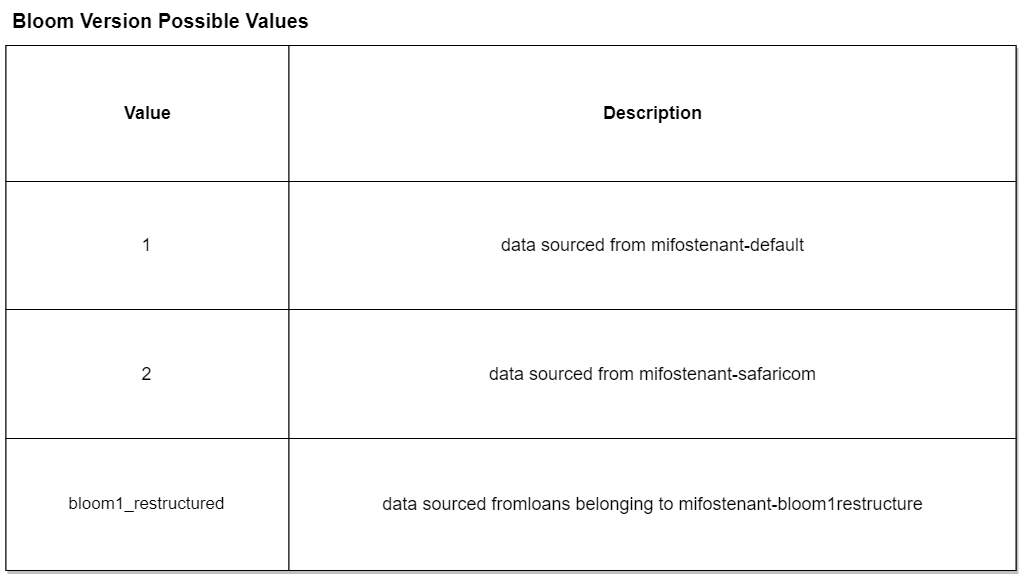
These views are present in all Kenyan product related schemas in the data warehouse.

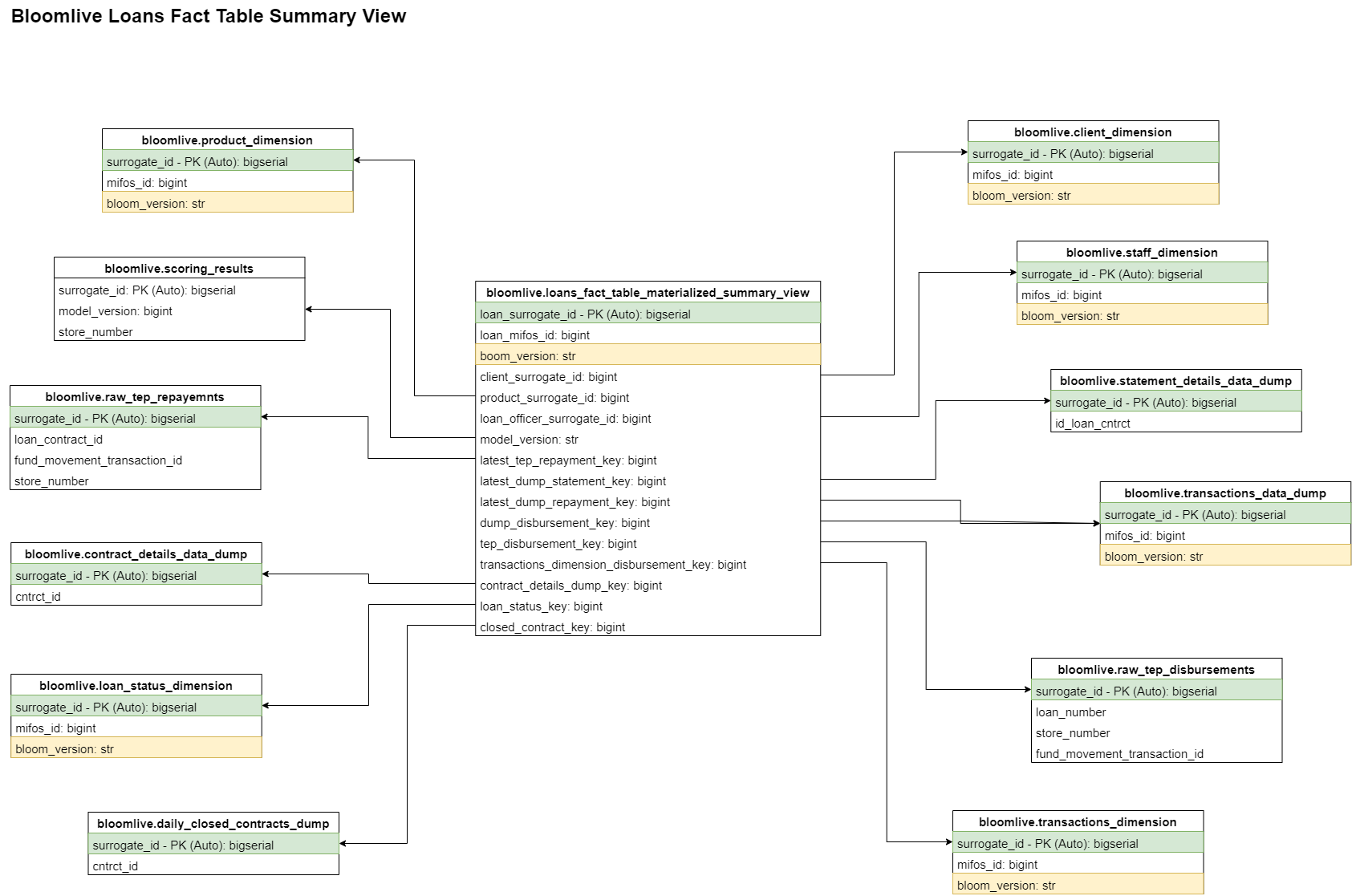
The “central\_bank\_of\_kenya” schema contains tables with the enum values provided by TransUnion for standard templating of data.

Business logic is captured in the source query for the respective views.

## Product: Safaricom Bloom







# CHAPTER 2: DATA PIPELINES

## Connections and Variables

Passwords and secrets used by the data pipelines are stored in Airflow variables and connections.

The values are encrypted in Fernet. More details on how to encrypt and decrypt values can be found in the [official Airflow documentation](https://airflow.apache.org/docs/apache-airflow/stable/security/secrets/fernet.html)

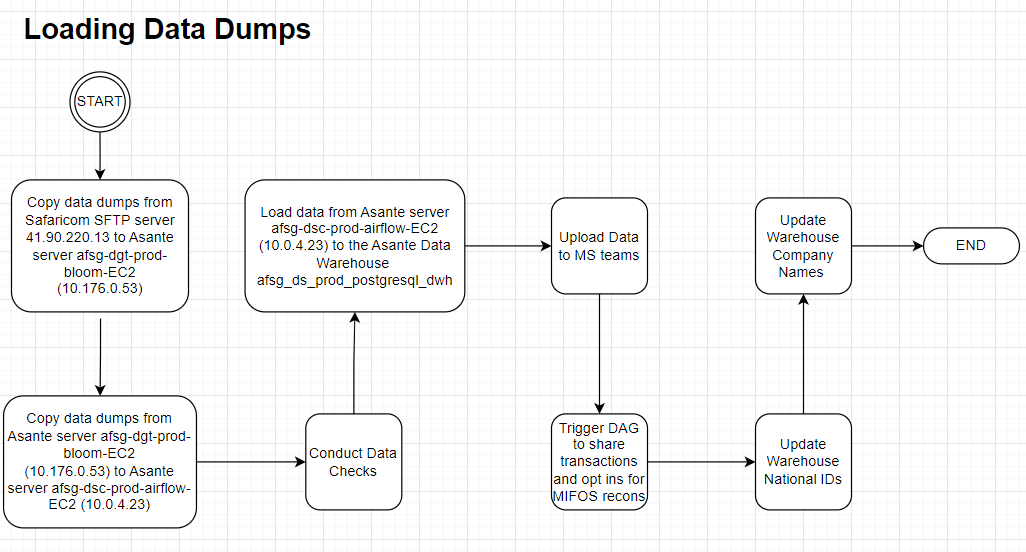
The Fernet key used by Asante FSG can be found in the /root/airflow/airflow.cfg file on server afsg-dsc-prod-airflow-EC2 server (10.0.4.23).

The value is in the key “fernet\_key” of the config file

## Product: Safaricom Bloom

### [DAG ID: ETL\_safaricom\_data\_dumps](https://airflow.asantefsg.com/data-pipelines/dags/ETL_safaricom_data_dumps/grid)

#### **Overview**



This data pipeline is designed to retrieve data from Safaricom SFTP (Simple File Transfer Protocol) server 41.90.220.13. The data is shared by Safaricom team in csv files. The data pipeline is scheduled to run daily at 11 am EAT.

This SFTP server is only accessible from Asante managed server “afsg-dgt-prod-bloom-EC2" of private IP 10.176.0.53. A VPN tunnel is configured between the two servers. A username and password is required to connect to the SFTP server.

8 files are expected daily by 11 am. Data is shared on a T-1 basis, for example, for the files below, they were received on 20230809:

1. daily\_active20230808.csv
2. daily\_closed\_contracts\_20230808.csv
3. daily\_contracts\_20230808.csv
4. daily\_deactive20230808.csv
5. daily\_disbursements20230808.csv
6. daily\_outst\_20230808.csv
7. daily\_repayment\_20230808.csv
8. statement\_details\_20230808.csv

The file naming convention is the dump name together with the date of the dump.

After the data is loaded into the data warehouse, the data is then shared on the MS Teams channel [Data Dumps.](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) This is made possible using the utils.office365api module which utilizes [Microsoft Graph API.](https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/ApplicationsListBlade)

The [data\_dumps\_bloom\_data\_recon](https://airflow.asantefsg.com/data-pipelines/dags/data_dumps_bloom_data_recon/grid) data pipeline Is also triggered

To monitor the data, a [Grafana](https://grafana.asantefsg.com/d/yUxy9pT7z/data-dumps?orgId=1) dashboard used.

Credentials for the SFTP server are stored in Apache Airflow variables; safaricom\_bloom\_sftp\_password, safaricom\_bloom\_sftp\_username and safaricom\_bloom\_sftp\_host.

The SFTP password expires every 30 days and must be reset by an Asante Team member on the next log in after expiration.

#### **Configuration Parameters.**

The DAG takes in one parameter; **files\_date**. If the parameter is not provided, the data pipeline defaults to T-1 date. This parameter is used to specify the date for which to fetch files from the server. For example, passing 20230105 as the value will fetch statement\_details\_20230105.csv and other files for that day.

{"files\_date":"20230105"}

#### **Tasks**

1. **purge\_previous\_files\_on\_airflow\_server**

This task forcefully deletes previously copied data dumps on the afsg-dsc-prod-airflow-EC2 server in the folder location specified by the “local\_files\_directory\_path” variable. It utilizes a BashOperator

1. **copy\_files\_from\_safaricom\_server\_to\_gateway\_server**

This task copies the files from the SFTP server to the server afsg-dgt-prod-bloom-EC2 in the folder specified by the remote\_files\_directory\_path variable. It utilizes an SSHOPerator with the ssh connection id “ssh\_gateway\_server”. After copying the files, it pushes the absolute paths of the copied files to the XCOM variable for use by subsequent tasks.

This is configured to retry 10 times before the DAG is marked as failed.

1. **copy\_files\_from\_gateway\_server\_to\_airflow\_server**

This task copies files from the afsg-dgt-prod-bloom-EC2 server to the “local\_files\_directory\_path” folder of the afsg-dsc-prod-airflow-EC2 server. It utilizes a custom Class that inherits from the SFTPOperator to retrieve files from the absolute paths pushed to the XCOM variable by “copy\_files\_from\_safaricom\_server\_to\_gateway\_server” task

1. **load\_contract\_details**

This task loads the daily\_contracts dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.  
The data is then checked for mandatory fields 'id\_date', 'cntrct\_id', 'src\_acces\_fee', 'src\_id\_idnty', 'src\_prncpl\_amnt'. An extra column ‘has\_missing\_fields’ is added to the raw data and set to True for any row that is missing data for the mandatory fields. For rows that have all mandatory fields populated, the ‘has\_missing\_fields’ column is marked as False.

After this step, the data is loaded into the contract\_details\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_contracts\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention contracts\_{file\_date}.csv. This function also removes any contract details dump duplicates in the data warehouse It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_daily\_closed\_contracts**

This task loads the daily\_closed\_contracts dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.  
The data is then checked for mandatory fields 'id\_date', 'cntrct\_id'. An extra column ‘has\_missing\_fields’ is added to the raw data and set to True for any row that is missing data for the mandatory fields. For rows that have all mandatory fields populated, the ‘has\_missing\_fields’ column is marked as False.

After this step, the data is loaded into the daily\_closed\_contracts\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_closed\_contracts\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention daily\_closed\_contracts\_{file\_date}.csv. This function also removes any daily closed contracts dump duplicates in the data warehouse. It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_daily\_disbursements**

This task loads the daily\_disbursements dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.

Two new columns; is\_repayment and is\_disbursement, are added to the raw data. The is\_repayment column is marked False for all rows while the is\_disbursement column is marked True for all rows.

The data is then checked for mandatory fields 'id\_loan\_cntrct', 'trxn\_amnt', 'id\_date', 'id\_trxn', 'cd\_mpsa\_orga\_shrt', 'trxn\_type', 'trxn\_stts', 'id\_trxn\_linkd',  
'id\_idnty', 'src\_assgnd\_crdt\_lmt', 'src\_used\_crdt\_lmit', 'src\_avail\_crdt\_lmit'.

An extra column ‘has\_missing\_fields’ is added to the raw data and set to True for any row that is missing data for the mandatory fields. For rows that have all mandatory fields populated, the ‘has\_missing\_fields’ column is marked as False.

After this step, the data is loaded into the transactions\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_disbursements\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention disbursements\_{file\_date}.csv. This function also removes any disbursement dump duplicates in the data warehouse. It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_daily\_repayments**

This task loads the daily\_repayments dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.

Two new columns; is\_repayment and is\_disbursement, are added to the raw data. The is\_repayment column is marked True for all rows while the is\_disbursement column is marked False for all rows.

The data is then checked for mandatory fields 'id\_date', 'id\_trxn', 'cd\_mpsa\_orga\_shrt', 'trxn\_type', 'trxn\_amnt', 'maintnanc\_fee', 'trxn\_stts', 'id\_trxn\_linkd', 'id\_idnty', 'id\_loan\_cntrct', 'dt\_trxn\_end'.

An extra column ‘has\_missing\_fields’ is added to the raw data and set to True for any row that is missing data for the mandatory fields. For rows that have all mandatory fields populated, the ‘has\_missing\_fields’ column is marked as False.

After this step, the data is loaded into the transactions\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_repayments\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention repayments\_{file\_date}.csv. This function also removes any repayments dump duplicates in the data warehouse. It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_daily\_active**

This task loads the daily\_active dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.

Two new columns; is\_opt\_in and is\_opt\_out, are added to the raw data. The is\_opt\_in column is marked True for all rows while the is\_opt\_out column is marked False for all rows.

The data is then checked for mandatory fields 'mpsa\_orga\_shrt', 'enty\_name', 'cust\_idnty\_id', 'cust\_id\_nmbr', 'nr\_mpsa\_enty\_phne', 'opt\_in\_date', 'ds\_mpsa\_enty\_name'.

An extra column ‘has\_missing\_fields’ is added to the raw data and set to True for any row that is missing data for the mandatory fields. For rows that have all mandatory fields populated, the ‘has\_missing\_fields’ column is marked as False.

After this step, the data is loaded into the client\_activity\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_opt\_ins\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention opt\_ins\_{file\_date}.csv. This function also removes any opt\_in dump duplicates in the data warehouse. This function in turn calls the upload\_all\_opt\_ins\_ms\_teams function that uploads all dump opt\_ins to the MS Teams channel with the name opt\_ins\_full\_{file\_date}.csv

It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_daily\_deactive**

This task loads the daily\_deactive dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.

Two new columns; is\_opt\_in and is\_opt\_out, are added to the raw data. The is\_opt\_in column is marked False for all rows while the is\_opt\_out column is marked True for all rows.

After this step, the data is loaded into the client\_activity\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_general\_opt\_outs function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention opt\_outs\_{file\_date}.csv. This function also removes any opt\_out dump duplicates in the data warehouse. It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_daily\_outstanding**

This task loads the daily\_outstanding dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.

After this step, the data is loaded into the outstanding\_loans\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_outstanding\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention outstanding\_{file\_date}.csv. This function also removes any daily\_outstanding dump duplicates in the data warehouse. It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **load\_statement\_details**

This task loads the statement\_details dump from the “local\_files\_directory\_path” folder into the data warehouse. It retrieves the file using the “files\_date” DAG parameter.

After this step, the data is loaded into the statement\_details\_data\_dump table of the bloomlive schema of the data warehouse.

After loading the data into the data warehouse, the upload\_statements\_ms\_teams function is called to upload the data to [Microsoft Teams](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?newTargetListUrl=%2Fsites%2FDataDumps%2FShared%20Documents&viewpath=%2Fsites%2FDataDumps%2FShared%20Documents%2FForms%2FAllItems%2Easpx&isAscending=false&id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) with the naming convention statement\_details\_{file\_date}.csv. This function also removes any statement\_details dump duplicates in the data warehouse. It is important to note that the data uploaded to MS Teams is fetched from the data warehouse.

1. **trigger\_data\_dumps\_bloom\_data\_recon**

This task utilize the trigger\_dag\_remotely function of the utils.common module of the Data Pipelines project to trigger the DAG [data\_dumps\_bloom\_data\_recon.](https://airflow.asantefsg.com/data-pipelines/dags/data_dumps_bloom_data_recon/grid)

1. **update\_warehouse\_national\_ids**

This task updates the national IDs of clients in the client dimension table of the bloomlive schema of the data warehouse. It updates Safaricom Bloom clients

A) With long-format national IDs in the warehouse

B) Without national IDs in the warehouse

After executing the updates, the logs are stored in the mysql database bloom\_pipeline hosted on afsg-dsc-prod-airflow-EC2 server, in the table national\_id\_updates with the src column populated by value “DUMPS”. The logs are stored by calling the store\_national\_id\_updates function of the utils.common module of the Data Pipelines project

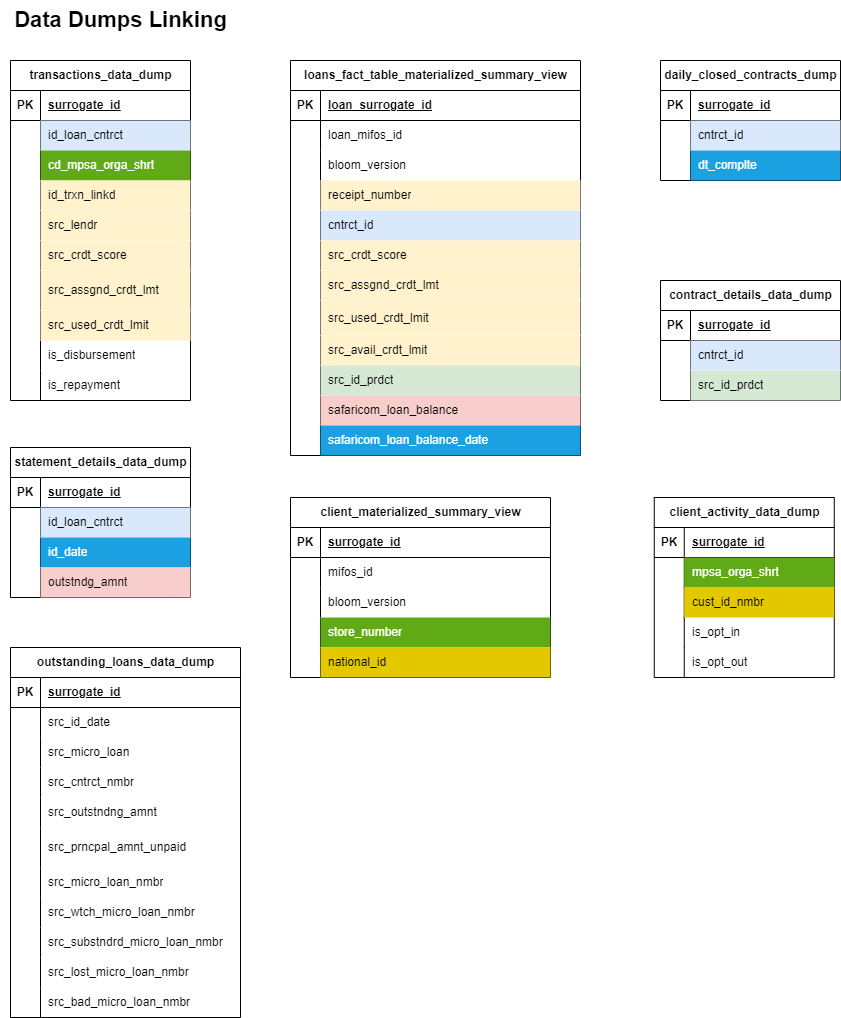
1. **update\_warehouse\_company\_names**

This task updates the national IDs of clients in the client dimension table of the bloomlive schema of the data warehouse. It updates Safaricom Bloom clients

A) With without company name in the data warehouse

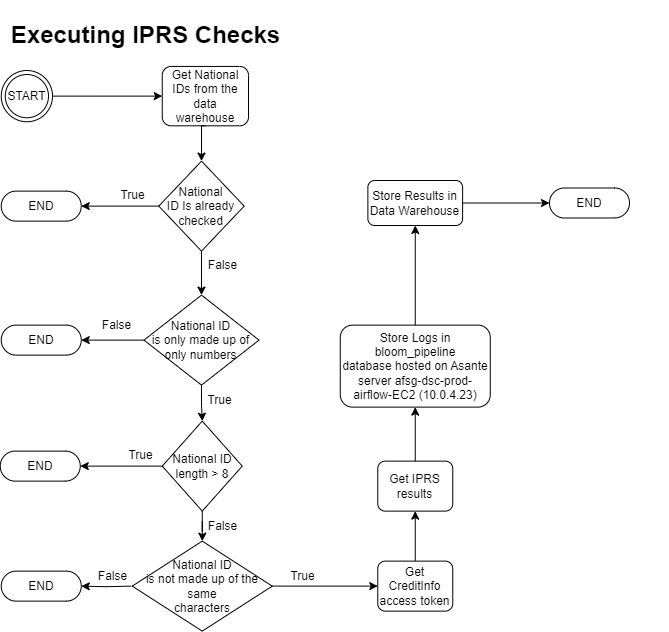
After executing the updates, the logs are stored in the mysql database bloom\_pipeline hosted on afsg-dsc-prod-airflow-EC2 server, in the table company\_name\_updates with the src column populated by value “DUMPS”. The logs are stored by calling the store\_company\_name\_updates function of the utils.common module of the Data Pipelines project

#### **Data**



### [DAG ID: Bloom\_IPRS\_check](https://airflow.asantefsg.com/data-pipelines/dags/Bloom_IPRS_check/grid)

#### **Overview**



This data pipeline is designed to fetch customer KYC information based on a Kenyan National ID from CreditInfo credit bureau service. The data pipeline makes use of the utils.creditInfo\_api module to make calls to the credit bureau service. This data pipeline is triggered remotely by the [ETL\_TEP\_data](https://airflow.asantefsg.com/data-pipelines/dags/ETL_TEP_data/grid) data pipeline

After retrieving the KYC information, the logs are stored in the database bloom\_pipeline hosted on Asante server afsg-dsc-prod-airflow-EC2 (10.0.4.23) in the table iprs\_idm\_logs. The unpacked data is stored in the warehouse table iprs\_kyc of the bloomlive schema of the data warehouse.

Credentials for production CreditInfo IPRS checks are stored in the Apache Airflow variables; idm\_live\_password, idm\_live\_username and iprs\_live\_strategy\_id

#### **Configuration Parameters.**

This data pipeline does not take in any parameters

#### **Tasks**

1. **get\_and\_refresh\_national\_ids**

This task retrieves national IDs that have not been passed through the IPRS check from the client\_materialized\_summary\_view of the bloomlive schema. This is based on the is\_iprs\_checked column of the materialized view.

The national IDs to be checked on IPRS must also pass three data checks;

1. The national ID must be made up of only numbers
2. The length of the national ID must not exceed 8 characters
3. The national ID must NOT be made up of the same characters e.g 00000000

The data pipeline iterates the national IDs in chunks of 5 and calls the API. This is to reduce occurrence of the “too many requests” error from CreditInfo API.

Calling the API is in two steps, first, the API gets the access token using the get\_token function of the utils.creditinfo\_api module of the Data Pipelines project. It retries getting the token two times. Upon receiving the access token, the second step is to get the actual KYC data from CreditInfo using the get\_results function of the utils.creditinfo\_api module of the Data Pipelines project. The second step is retried infinitely id the response status code from CreditInfo is 202 I.e “too many requests” with a 2 second delay between retries.

If the request is successful, the ‘FirstName’, ‘Surname', 'OtherName', 'DateOfIssue' and ‘SerialNumber’portions of the data are unpacked.

After making calls to CreditInfo API, the task stores the raw response in the bloom\_pipeline database hosted on afsg-dsc-prod-airflow-EC2 server in the table iprs\_idm\_logs.

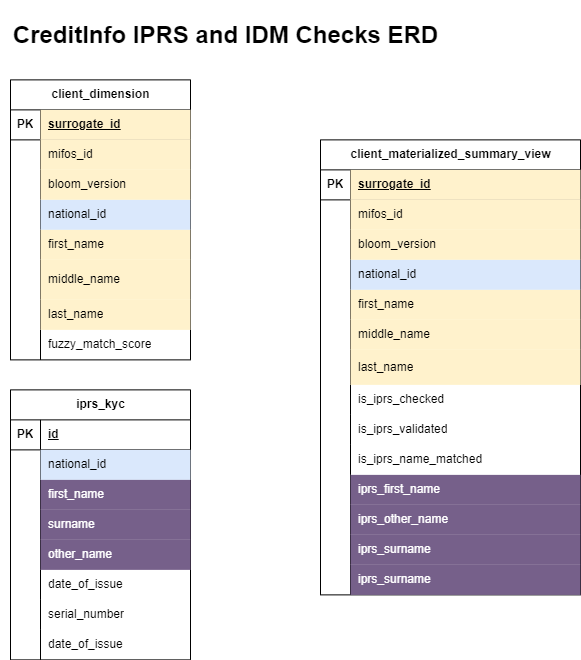
The task then proceeds to store the unpacked data in the iprs\_kyc table of the bloomlive schema of the data warehouse.

1. **get\_fuzzy\_match\_scores\_bloom\_client\_dimension**

This task calculates the similarity score between the iprs names and MIFOS names in the data warehouse for the client\_dimension table of the bloomlive schema of the data warehouse. The score is calculated only for those rows in the client\_dimension table that have the value for fuzzy\_match\_score column as null.

Once the score is calculated, the score is updated in the data warehouse.

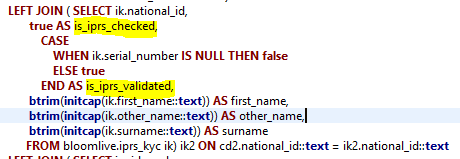
#### **Data**



**Data Summary**

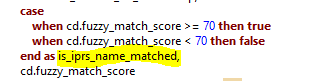
The column is\_iprs\_checked is a boolean column that is True if the national id of a client is in the iprs\_kyc table. The column i**s\_iprs\_validated** is a boolean column that is True if the **serial\_number** for a national id is present in the bloomlive.iprs\_kyc table. A successful iprs call returns a serial number for the national id.

Below is a section of the query that results in the client\_summary\_view view. It shows how the boolean values for is iprs\_checked column and is\_iprs\_validated column are populated. cd2 represents the client dimension table.



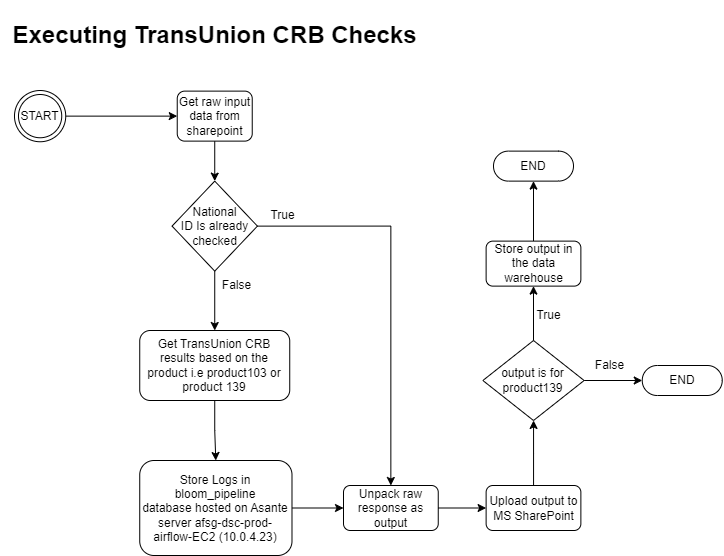
The column is\_iprs\_name\_matched is a boolean column that is True if the fuzzy\_match\_score column has a value greater than or equal to 70 and False if the fuzzy\_match\_score column has a value less than 70. The fuzzy\_match\_score column value is determined by calculating the similarity score between the iprs names and mifos names. This is done in the “get\_fuzzy\_match\_scores\_bloom\_client\_dimension” step of the Bloom\_IPRS\_Check DAG.

Below is a section of the query that results in the client\_summary\_view view. It shows how the boolean value for is iprs\_name\_matched is populated. cd represents the client dimension table.



### [DAG ID: TransUnion\_CRB\_checks](https://airflow.asantefsg.com/data-pipelines/dags/TransUnion_CRB_checks/grid)

#### **Overview**



This data pipeline is designed to retrieve credit bureau data from TransUnion CRB. The data pipeline is configured to fetch data for product139 (High Velocity Check) and product103 (Skip Trace).

The data pipeline is triggered manually.

The DAG expects the raw input file to have the following structure.

1. It must be a valid Excel file (.xlsx or .xls).
2. It must contain the following columns:
   1. national\_id: The national ID of the individual (Mandatory).
   2. first\_name: The first name of the individual.
   3. middle\_name: The middle name of the individual.
   4. surname: The surname or last name of the individual.
   5. mobile\_number: The mobile number of the individual (optional).
   6. product\_name: The product name from TransUnion. Currently, the DAG only accepts two options; 'product139' or 'product103'.

For each record in the input file, the record must have at least two names of the expected 3 names (first\_name, middle\_name, last\_name). If not provided, the API calls will fail.

The DAG stores raw logs in the database server 10.0.4.23 in the table "transunion\_crb\_checks"

The DAG stores cleaned data in [the Output folder](https://netorgft4232141.sharepoint.com/:f:/s/SafaricomBloom/EkYvhybsicJCht6on4Eci30BdffzM_i6SBB9SgwRXhVz4g?e=9jxqfj) in a csv file

The output file is named the same as the input file but with the word 'out' appended to it for example, if the input file is named 'Skip\_Trace\_2023\_08\_09.xlsx', the output file will be named 'Skip\_Trace\_2023\_08\_09\_out.csv'

If the product was "product139" then the cleaned data is also stored in the data warehouse. This is in the schema bloomlive, in the table transunion\_high\_velocity\_crb\_checks

#### **Configuration Parameters.**

The pipeline takes in two parameters; file\_name and product\_name. The product\_name parameter can only be one of two options; **"product139"** or **"product103",** where "product139" is the High Velocity product while "product103" is the Skip Trace product.

The product name is **case sensitive**.

The file\_name is the name of a file in [the Input folder](https://netorgft4232141.sharepoint.com/:f:/s/SafaricomBloom/Emr2JRC6M-ZHn5Y-w415M8IBDEmzoYpGfbPEeCcoF0tlww?e=CFSwHr) of the 'Safaricom Bloom' site and should include the file extension.

Examples;

{"file\_name": "Skip\_Trace\_2023\_08\_09.xlsx", "product\_name": "product103"}  
  
{"file\_name": "Bloom Merchants With Limits.xlsx", "product\_name": "product139"}

**API Call Logic**

To determine whether to retrieve data from the logs or call the TransUnion API, consider the following scenarios:

1. **Same File Name, Same Product:**

* If you run the same file with the same product as before, the results will be fetched from the existing logs for only previously successful requests. For previously failed requests and new national IDs in the file, the TransUnion API will be called for result retrieval

1. **Same File Name, Different Product:**

* Running the same file with a different product will trigger a call to the TransUnion API for result retrieval.

1. **Different File Name, Same National IDs:**

* If you use a different file containing the same national IDs, the DAG will initiate a request to the TransUnion API.

#### **Tasks**

1. **get\_credit\_report**

This task is responsible for making the API calls to Transunion. The task retrieves the input file from sharepoint using the file\_name DAG parameter. To determine the product, I.e. High velocity or Skip Trace, the task relies on the product\_name DAG parameter.

After retrieving the input file, the task checks for each national\_id whether they have already been queried before. This is based on below query

The task removes national ids in the input data that have already been checked before successfully. If there are remaining national IDs, the task proceeds to query the TransUnion API using the utils.transunion\_api module of the Data Pipelines project.

Each response is stored in the transunion\_crb\_logs table of the bloom\_pipeline mysql database hosted on 10.0.4.23 server.

1. **unpack\_raw\_responses**

This task is responsible for unpacking and flattening the raw responses from the logs table. To determine the logs to retrieve cleaned data, the task relies on the file\_name and product\_name parameters of the DAG.

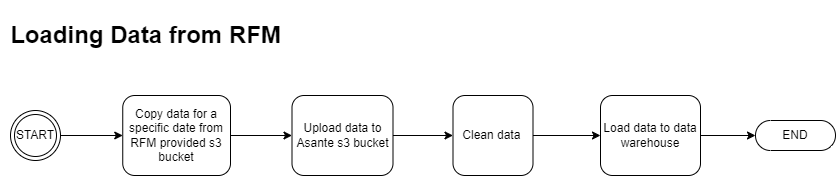
The task retrieves the raw responses from the logs database and unpacks successful requests only.



After unpacking and flattening the data, the task calls the store\_results function. The function uploads the file into the [output sharepoint folder.](https://netorgft4232141.sharepoint.com/sites/SafaricomBloom/Shared%20Documents/Forms/AllItems.aspx?FolderCTID=0x01200088E2DCAF8B383C489F826D5AFC7A573D&isAscending=false&id=%2Fsites%2FSafaricomBloom%2FShared%20Documents%2FTransunion%20Credit%20Reports%2FOutput&sortField=Modified&viewid=aff5ca06%2D3e55%2D414b%2D9b83%2De92efb25265f) If the product is High Velocity, the function stores the unpacked data into the transunion\_high\_velocity\_crb\_checks table of the bloomlive schema of the data warehouse.

### [DAG ID: ETL\_bloom\_retail\_field\_metrics](https://airflow.asantefsg.com/data-pipelines/dags/ETL_bloom_retail_field_metrics/grid)

#### **Overview**



This data pipeline is designed to fetch retail field metrics data from the partner s3 bucket and loads it into the data warehouse. The data is sourced from Asante FSG field agents.

The DAG is scheduled to run daily at 11 am.

#### **Configuration Parameters.**

The data pipeline takes in an optional parameter, “files\_date”. If the parameter is not provided, the data pipeline defaults to using the current date as the value of the parameter.

This parameter is used to fetch files for a specific date.

#### **Tasks**

1. **Copy\_files\_from\_s3\_bucket\_to\_warehouse**

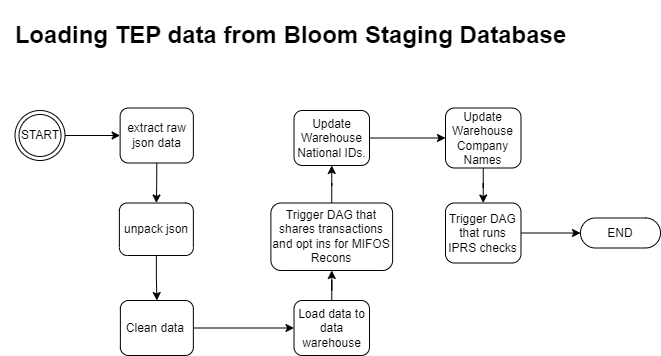
This task searches the partner s3 bucket for any files which have a name containing the value of the “files\_date” parameter. Upon finding files, the task copies the file from the partner s3 bucket and uploads it into [Asante FSG S3 bucket.](https://s3.console.aws.amazon.com/s3/buckets/afsg-ds-prod-postgresql-dwh-archive?region=eu-central-1&prefix=safaricom_bloom/retail_field_metrics/&showversions=false)

Credentials for the partner s3 bucket are stored in Airflow variables: “RFM\_aws\_access\_key\_id” and “RFM\_aws\_secret\_access\_key”

After uploading the data to the s3 bucket, the task cleans the raw data and stores it in the retail\_field\_metrics table of the bloomlive schema of the data warehouse.

### [DAG ID: ETL\_TEP\_data](https://airflow.asantefsg.com/data-pipelines/dags/ETL_TEP_data/grid)

#### **Overview**

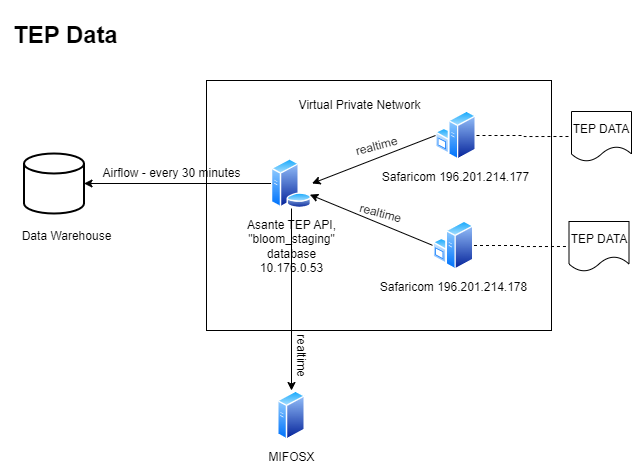


This data pipeline is designed to load TEP (Transaction Event Publisher) data from bloom\_staging2 database to the data warehouse. It is scheduled to run every 30 minutes.

TEP data is streamed from Safaricom and contains disbursements, repayments, opt ins and opt outs.

The bloom\_staging2 database is the application database for the TEP API endpoint which is maintained by Digital Team.

The bloom\_staging2 database stores TEP data as it is before being posted to MIFOSX and also the data as it is after posting to MIFOSX. Data that has been posted to MIFOSX successfully has the column “status\_code” populated by 200. If the posting was unsuccessful, the column “status\_code” is populated by the value 400.



#### **Configuration Parameters.**

The ETL\_TEP\_data data pipeline does not take in any configuration parameters.

#### **Tasks**

1. **extract\_repayments**

This task extracts repayments data from bloom\_staging2 database. It extracts transactions that occurred not more than 30 days from the current date.

The task iterates the transactions in chunks of 100000 and unpacks json data from the raw\_request column.

After cleaning the data, the task checks for already existing records in the raw\_tep\_repayments table of the bloomlive schema of the data warehouse. If there are existing records, the task updates the status codes and failure reasons in the data warehouse for the existing records.

The task proceeds to store only new data in the data warehouse.

1. **extract\_disbursements**

This task extracts disbursements data from bloom\_staging2 database. It extracts transactions that occurred not more than 30 days from the current date.

The task iterates the transactions in chunks of 100000 and unpacks json data from the raw\_request column.

After cleaning the data, the task checks for already existing records in the raw\_tep\_disbursements table of the bloomlive schema of the data warehouse. If there are existing records, the task updates the status codes and failure reasons in the data warehouse for the existing records.

The task proceeds to store only new data in the data warehouse.

1. **extract\_opt\_ins**

This task extracts opt-in data from bloom\_staging2 database. It extracts opt-ins that occurred not more than 30 days from the current date.

The task iterates the transactions in chunks of 100000 and unpacks json data from the raw\_request column.

After cleaning the data, the task checks for already existing records in the raw\_tep\_client\_activity table of the bloomlive schema of the data warehouse. If there are existing records, the task updates the status codes and failure reasons in the data warehouse for the existing records.

The task proceeds to store only new data in the data warehouse.

1. **extract\_opt\_outs**

This task extracts opt-out data from bloom\_staging2 database. It extracts opt- outs that occurred not more than 30 days from the current date.

The task iterates the transactions in chunks of 100000 and unpacks json data from the raw\_request column.

After cleaning the data, the task checks for already existing records in the raw\_tep\_client\_activity table of the bloomlive schema of the data warehouse. If there are existing records, the task updates the status codes and failure reasons in the data warehouse for the existing records.

The task proceeds to store only new data in the data warehouse.

1. **trigger\_tep\_data\_recon**

This task uses the trigger\_dag\_remotely function of the utils.common module to trigger the [tep\_data\_bloom\_recon](https://airflow.asantefsg.com/data-pipelines/dags/tep_data_bloom_recon/grid) data pipeline

1. **update\_warehouse\_national\_ids**

This task updates the national IDs of clients in the client dimension table of the bloomlive schema of the data warehouse. It updates Safaricom Bloom clients

A) With long-format national IDs in the warehouse

B) Without national IDs in the warehouse

After executing the updates, the logs are stored in the mysql database bloom\_pipeline hosted on afsg-dsc-prod-airflow-EC2 server, in the table national\_id\_updates with the src column populated by value “TEP”. The logs are stored by calling the store\_national\_id\_updates function of the utils.common module of the Data Pipelines project

1. **update\_warehouse\_company\_names**

This task updates the national IDs of clients in the client dimension table of the bloomlive schema of the data warehouse. It updates Safaricom Bloom clients

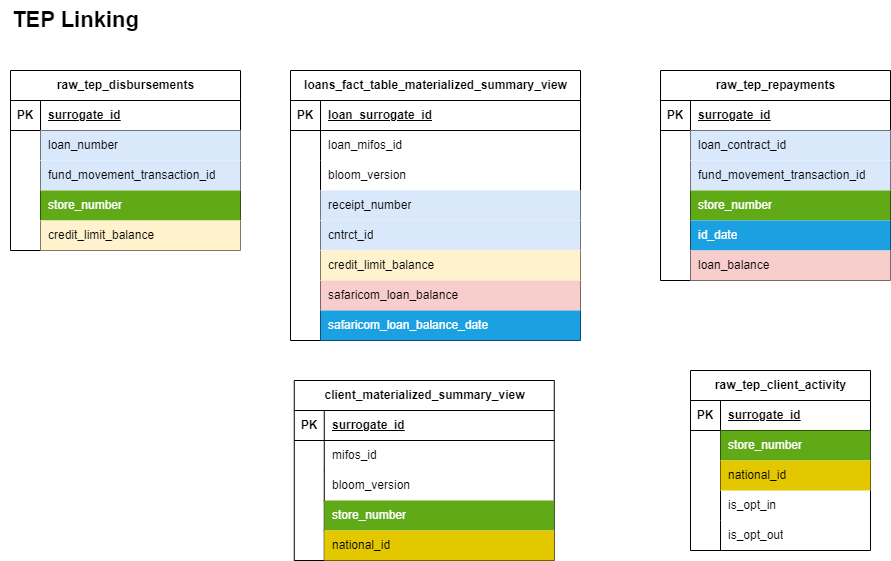
A) With without company name in the data warehouse

After executing the updates, the logs are stored in the mysql database bloom\_pipeline hosted on afsg-dsc-prod-airflow-EC2 server, in the table company\_name\_updates with the src column populated by value “TEP”. The logs are stored by calling the store\_company\_name\_updates function of the utils.common module of the Data Pipelines project

1. **Trigger\_Bloom\_IPRS\_check**

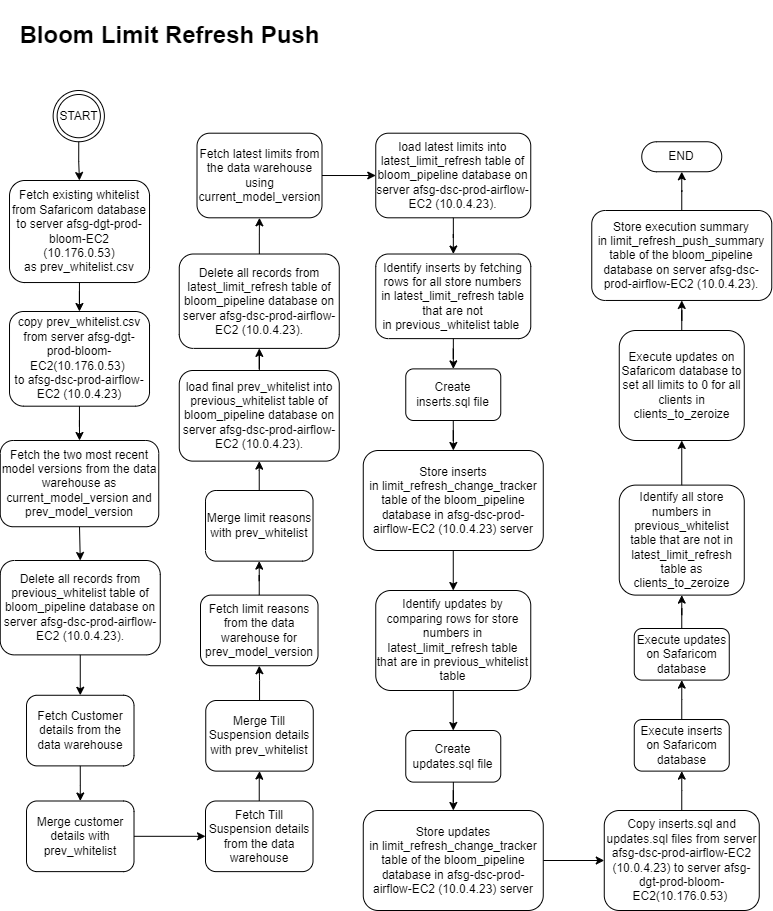
This task uses the trigger\_dag\_remotely function of the utils.common module to trigger the [Bloom\_IPRS\_check](https://airflow.asantefsg.com/data-pipelines/dags/Bloom_IPRS_check/grid) data pipeline

#### **Data**

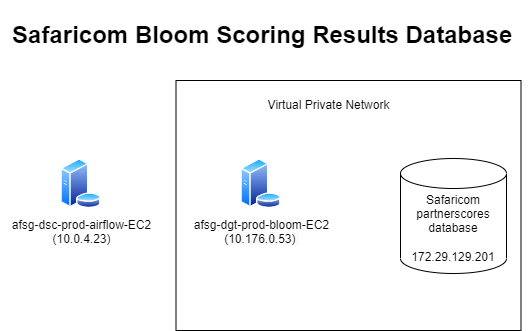


### [DAG ID: Bloom\_limit\_refresh\_push](https://airflow.asantefsg.com/data-pipelines/dags/Bloom_limit_refresh_push/grid)

#### **Overview**



This data pipeline is designed to push Safaricom Bloom scoring results into Safaricom database. Whilst achieving this, the data pipeline also stores the whitelist on safaricom database as it is before being updated, for purposes of customer engagement.



The data pipeline is triggered via API from the data scientists scoring scripts.

Credentials for Safaricom database connection are stored in the airflow variables: safaricom\_bloom\_partner\_scoring\_results\_database\_username, safaricom\_bloom\_partner\_scoring\_results\_database\_password, safaricom\_bloom\_partner\_scoring\_results\_database\_host, safaricom\_bloom\_partner\_scoring\_results\_database\_port, safaricom\_bloom\_partner\_scoring\_results\_database\_name

#### **Configuration Parameters.**

The DAG takes in one parameter, model\_version. This parameter value should be the model\_version of the scoring results to be pushed to Safaricom database.

Below is an example to push scoring results whose model\_version is "2023-014[2023-07-04, 2023-08-18]"

**{**"model\_version": "2023-014[2023-07-04, 2023-08-18]"**}**

#### **Tasks**

1. **get\_previous\_whitelist**

This task extracts scoring results from the safaricom database dbo.PltAsanteFinanceList. It does this using an SSHOperator with the connection id ssh\_gateway\_server. The SSHOperator makes use of an sqlcmd command to extract the scoring results to a file prev\_whitelist.csv in the afsg-dgt-prod-bloom-EC2 (10.176.0.53) server. The file is stored in the location /root/data/safaricom\_bloom/limit\_refresh

1. **copy\_previous\_whitelist\_from\_gateway\_server\_to\_airflow\_server**

This task copies prev\_whitelist file generated by task 1 from server afsg-dgt-prod-bloom-EC2 (10.176.0.53) to server afsg-dsc-prod-airflow-EC2 (10.0.4.23). It makes use of an SFTP operator with the connection id ssh\_gateway\_server.

The prev\_whitelist.csv file is stored in the location /root/airflow/DataPipelines/data/safaricom\_bloom/limit\_refresh

1. **upload\_previous\_whitelist**

This task adds metadata to the prev\_whitelist.csv data and uploads it to the previous\_whitelist table of the bloom\_pipeline database hosted on the server afsg-dsc-prod-airflow-EC2 (10.0.4.23).

The task achieves this by first deleting all records in the previous\_whitelist table. The task then fetches client details, till suspension details and limit reasons from the data warehouse using the get\_bloom\_whitelist function of the utils.common module. It merges the data with prev\_whitelist data to form an enriched dataset.

This dataset is then inserted into the previous\_whitelist table. The customers and engagement team consumes this data via Grafana.

The task also deletes all records from latest\_limit\_refresh table of the bloom\_pipeline database. It then fetches the latest scoring results from the scoring\_results\_view of the bloomlive schema of the data warehouse and loads the data into the latest\_limit\_refresh table of the bloom\_pipeline database

1. **get\_data\_to\_be\_inserted**

This task compares data in the previous\_whitelist table and latest\_limit\_refresh table to identify records in the latest limit refresh that were not in the previous limit refresh. These are the inserts.

For each insert, an insert query is formulated for the safaricom database. The query also includes the columns “CreatedOn\_Date” and “ModifiedOn\_Date” set as the timestamp as at the time of generation of the query.

The queries are compiled and stored in a csv file with the naming format inserts\_{inserts\_file\_timestamp}.sql whereby the inserts\_file\_timestamp is the timestamp as at generation of the file. The file is stored in the location /root/airflow/DataPipelines/data/safaricom\_bloom/limit\_refresh of the afsg-dsc-prod-airflow-EC2 (10.0.4.23) server

If no inserts are found, no queries are formulated

After generating the file, the task stores the inserts into the table limit\_refresh\_change\_tracker of the bloom\_pipeline database.

Finally, the task pushes the inserts file name into an xcom variable for retrieval by other tasks in the DAG. The count of total inserts is also pushed into an xcom variable

1. **get\_data\_to\_be\_updated**

This task compares data in the previous\_whitelist table and latest\_limit\_refresh table to identify records in the latest limit refresh that were in the previous limit refresh which have changed limits.

For each column that changes, an update query is formulated. The update query also includes an update for the “ModifiedOn\_Date” column of the safaricom database which is populated by the timestamp as at the time of generation of the update query.

If no updates are found, no queries are formulated

The queries are compiled and stored in a csv file with the naming format updates\_{updates\_file\_timestamp}.sql whereby the updates\_file\_timestamp is the timestamp as at generation of the file. The file is stored in the location /root/airflow/DataPipelines/data/safaricom\_bloom/limit\_refresh of the afsg-dsc-prod-airflow-EC2 (10.0.4.23) server.

After generating the file, the task stores the updates into the table limit\_refresh\_change\_tracker of the bloom\_pipeline database.

Finally, the task pushes the updates file name into an xcom variable for retrieval by other tasks in the DAG. The count of total updates is also pushed into an xcom variable

1. **copy\_inserts\_and\_updates\_files\_from\_airflow\_server\_to\_gateway\_server**

This task copies the files: updates\_{updates\_file\_timestamp}.sql and inserts\_{inserts\_file\_timestamp}.sql, to the server afsg-dgt-prod-bloom-EC2 (10.176.0.53).

The task retrieves the file names from the xcom variable pushed by the respective tasks that generate the files.

If no updates were identified by previous tasks no update file is copied. If no inserts were identified by previous tasks no insert file is copied.

The files are stored in the location /root/data/safaricom\_bloom/limit\_refresh of the afsg-dgt-prod-bloom-EC2 (10.176.0.53) server

1. **execute\_inserts**

This task makes use of an SSHOperator to run the queries in the inserts\_{inserts\_file\_timestamp}.sql file on the Safaricom database using sqlcmd.

The SSHOperator makes use of the airflow connection ssh\_gateway\_server.

If no inserts were identified in previous tasks, the message "There are 0 inserts to make" is logged

1. **execute\_updates**

This task makes use of an SSHOperator to run the queries in the updates\_{updates\_file\_timestamp}.sql file on the Safaricom database using sqlcmd.

The SSHOperator makes use of the airflow connection ssh\_gateway\_server.

If no updates were identified in previous tasks, the message "There are 0 updates to make" is logged

1. **get\_clients\_to\_zeroize**

This task compares data in the previous\_whitelist table with data in the latest\_limit\_refresh table to identify records in the previous whitelist that are exempted from the latest limit refresh.

For any records identified, the task executes a query on the Safaricom database to zeroize all limits. The modified date is also modified to the timestamp as at the time of the update.

The count of total clients zerozed is also pushed into an xcom variable

This task is an error handling mechanism as ideally no records should be in the previous whitelist but missing in the latest limit refresh.

1. **send\_limit\_refresh\_push\_summary**

This task shares a notification to the “[Data Science Team](https://netorgft4232141.sharepoint.com/sites/DSGrafanaTest/Shared%20Documents/General?CT=1692705807441)” Microsoft Teams channel. The message shows the model version pushed and the total inserts, total updates and total zeroized clients.

1. **store\_execution\_summary**

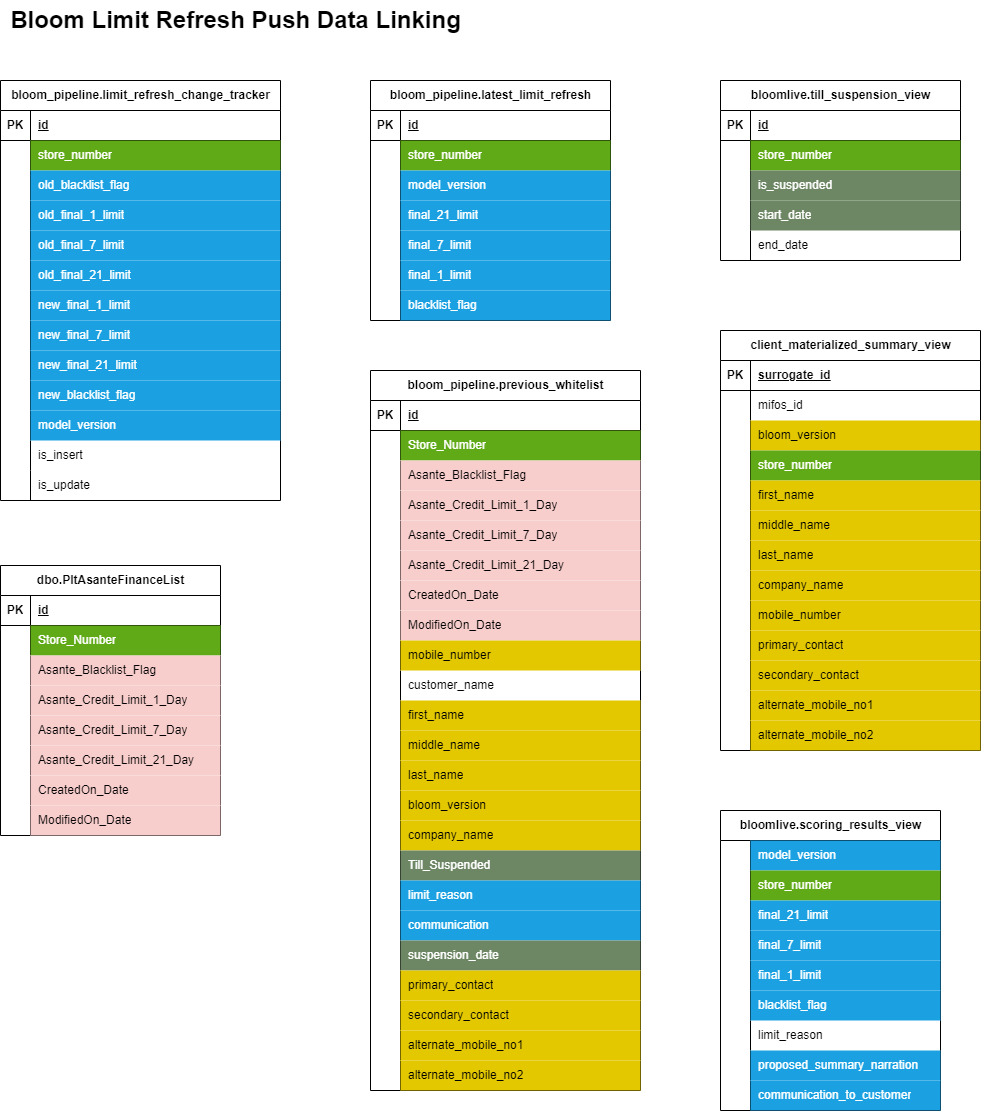
This task stores the model\_version, total\_inserts, total\_updates, the DAG start time as start\_date and the current timestamp as end\_date in the table limit\_refresh\_push\_summary of the bloom\_pipeline database.

This data is used by the [ETL\_bloom2\_loans\_fact\_table](https://airflow.asantefsg.com/data-pipelines/dags/ETL_bloom2_loans_fact_table/grid) data pipeline to update the loans fact table model versions

1. **trigger\_EL\_live\_bloom\_whitelist**

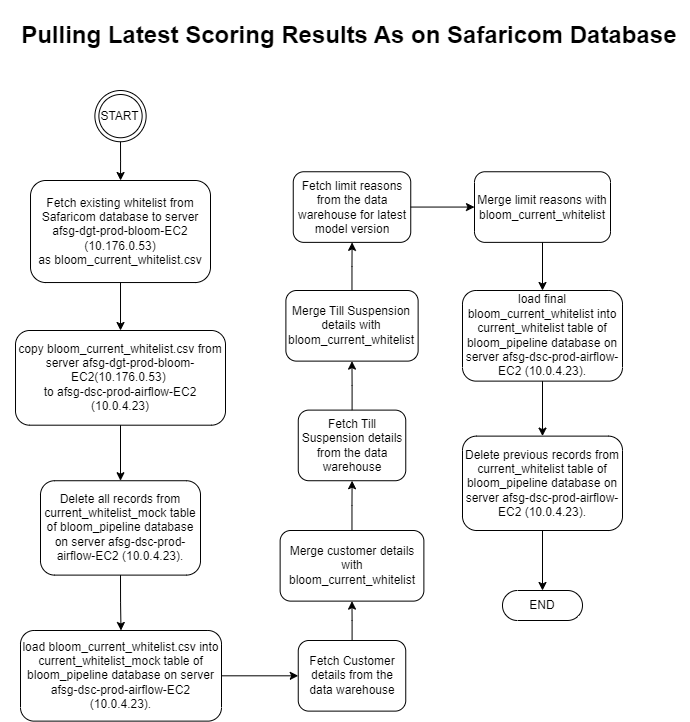
This is the last task of the DAG and it triggers the [EL\_live\_bloom\_whitelist](https://airflow.asantefsg.com/data-pipelines/dags/EL_live_bloom_whitelist/grid) data pipeline.

#### **Data**



### [DAG ID: EL\_live\_bloom\_whitelist](https://airflow.asantefsg.com/data-pipelines/dags/EL_live_bloom_whitelist/grid)

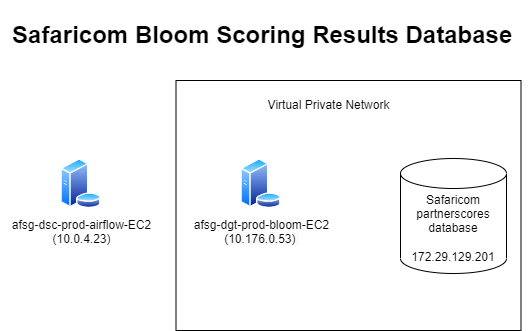
#### **Overview**



This data pipeline is designed to fetch the scoring results for Safaricom Bloom from Safaricom database. The data fetched by this data pipeline is used by the data scientist to confirm the results are as scored internally.

The data is also used by the Customer Engagement team while calling customers to inform them of the limits they qualify for.

The data pipeline is scheduled to run daily at noon.



#### **Configuration Parameters.**

This DAG does not take in any configuration parameters.

#### **Tasks**

1. **copy\_current\_whitelist\_from\_safaricom\_server\_to\_gateway\_server**

This task extracts scoring results from the safaricom database dbo.PltAsanteFinanceList. It does this using an SSHOperator with the connection id ssh\_gateway\_server. The SSHOperator makes use of an sqlcmd command to extract the scoring results to a file bloom\_current\_whitelist.csv in the afsg-dgt-prod-bloom-EC2 (10.176.0.53) server. The file is stored in the location /root/data/safaricom\_bloom/limit\_refresh/bloom\_current\_whitelist.csv

1. **copy\_files\_from\_gateway\_server\_to\_airflow\_server**

This task copies bloom\_current\_whitelist.csv file generated by task 1 from server afsg-dgt-prod-bloom-EC2 (10.176.0.53) to server afsg-dsc-prod-airflow-EC2 (10.0.4.23). It makes use of an SFTP operator with the connection id ssh\_gateway\_server.

The bloom\_current\_whitelist.csv file is stored in the location /root/airflow/DataPipelines/data/safaricom\_bloom/limit\_refresh/bloom\_current\_whitelist.csv

1. **save\_current\_whitelist**

This task saves the bloom\_current\_whitelist.csv file as it is on current\_whitelist\_mock table of the bloom\_pipeline database. The task also adds metadata to the bloom\_current\_whitelist.csv data and uploads it to the current\_whitelist table of the bloom\_pipeline database.

The bloom\_pipeline database is hosted on server afsg-dsc-prod-airflow-EC2 (10.0.4.23).

The task achieves this by first deleting all records in the current\_whitelist\_mock table and loading the bloom\_current\_whitelist.csv file as it is on the table .

The task then fetches client details, till suspension details and limit reasons from the data warehouse using the get\_bloom\_whitelist function of the utils.common module. It merges the data with bloom\_current\_whitelist.csv data to form an enriched dataset.

This dataset is then inserted into the current\_whitelist table. The customers and engagement team consumes this data via Grafana.

The task then deletes all previous records from current\_whitelist table of the bloom\_pipeline database.

1. **send\_current\_whitelist\_saved\_ms\_teams\_notification**

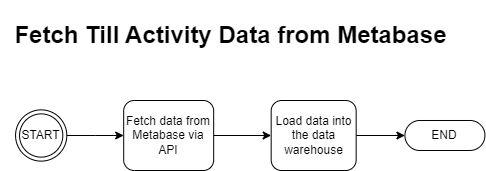
This task sends an MS Teams notification to the channel “[Data Science Team](https://netorgft4232141.sharepoint.com/sites/DSGrafanaTest/Shared%20Documents/General?CT=1692705807441)”, to notify that the DAG has run successfully

#### **Data**



### [DAG ID: ELT\_bloom\_till\_activity\_raw\_transactions](https://airflow.asantefsg.com/data-pipelines/dags/ELT_bloom_till_activity_raw_transactions/grid)

#### **Overview**



This data pipeline is designed to fetch data from Metabase via API using the utils.metabase\_api module of the DataPipelines project. The data pipeline loads the Fetched data into the data warehouse.

#### **Configuration Parameters.**

This data pipeline takes in the configuration parameter “data\_date”. It is not mandatory for this parameter to be provided.

If not provided, the DAG defaults to t-1 date as the value for the parameter.  
  
Example  
{"data\_date": "2023-01-05"}

#### **Tasks**

1. **extract\_load\_payments**

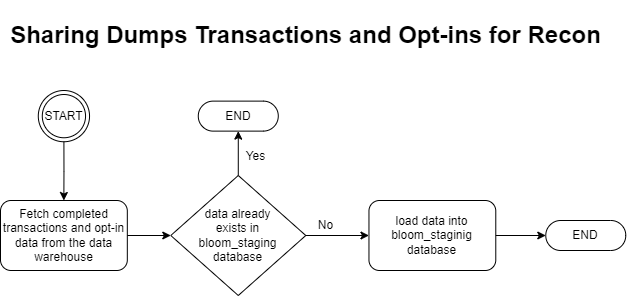
This task fetches data from Metabase using the utils.metabase\_api module. The task checks whether the data\_date parameter has been provided and if not provided, the task sets the value of data\_date to the current date minus one day date.

So as not to overload the API, the task iterates each hour of the specified date and fetches the data.

The data for each hour of the date provided is loaded in the bloomlive.till\_activity\_dimension table of the data warehouse.

### [DAG ID: data\_dumps\_bloom\_data\_recon](https://airflow.asantefsg.com/data-pipelines/dags/data_dumps_bloom_data_recon/grid)

#### **Overview**



This data pipeline is designed to load dumps data that is not already posted on MIFOSX into bloom\_staging database. The DAG is triggered by the [ETL\_safaricom\_data\_dumps](https://airflow.asantefsg.com/data-pipelines/dags/ETL_safaricom_data_dumps/grid)DAG**.**

#### **Configuration Parameters.**

This DAG does not take in any configuration parameters.

#### **Tasks**

1. **share\_disbursements**

This task fetches dump disbursements data from bloomlive.transactions\_data\_dump table of the data warehouse and loads it into transactions\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task only fetches those disbursements dumps that have an associated contract dump. This is achieved by merging data from bloomlive.transactions\_data\_dump table with data from bloomlive.contract\_details\_data\_dump table.

The task iterates over the fetched data and checks on bloom\_staging.transactions\_data\_dump table whether the records exist.

For records that do not exist in the target database, the task standardizes the data and loads them into the bloom\_staging.transactions\_data\_dump

1. **share\_repayments**

This task fetches dump repayments data from bloomlive.transactions\_data\_dump table of the data warehouse and loads it into transactions\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task iterates over the fetched data and checks on bloom\_staging.transactions\_data\_dump table whether the records exist.

For records that do not exist in the target database, the task fetches the corresponding disbursements receipt numbers from the data warehouse and merges with the data.

The data is further cleaned and standardized to fit into the bloom\_staging.transactions\_data\_dump table.

The data is then loaded into the bloom\_staging.transactions\_data\_dump table

1. **share\_opt\_ins**

This task fetches dump opt-in data from bloomlive.client\_activity table of the data warehouse and loads it into opt\_ins\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task iterates over the fetched data and checks on bloom\_staging.opt\_ins\_data\_dump table whether the records exist.

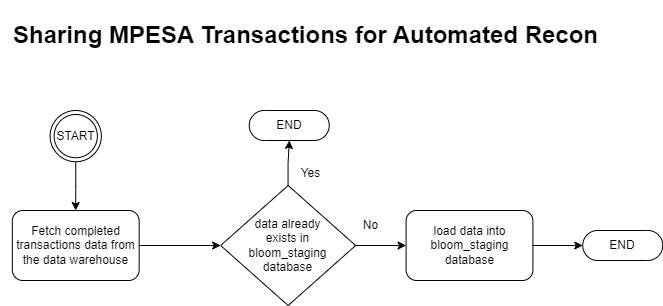
For records that do not exist in the target database, the task standardizes the data and loads them into the bloom\_staging.opt\_ins\_data\_dump

1. **trigger\_ETL\_safaricom\_recons\_mifos\_posting**

This task triggers the [ETL\_safaricom\_recons\_mifos\_posting](https://airflow.asantefsg.com/data-pipelines/dags/ETL_safaricom_recons_mifos_posting/grid) DAG

### [DAG ID: mpesa\_statements\_bloom\_data\_recon](https://airflow.asantefsg.com/data-pipelines/dags/mpesa_statements_bloom_data_recon/grid)

#### **Overview**



This data pipeline is designed to load raw mpesa disbursements and repayments data that is not already posted on MIFOSX into bloom\_staging database. Raw mpesa transactions data is provided by the Operations Team in the SharePoint site [Safaricom Bloom - Home (sharepoint.com)](https://netorgft4232141.sharepoint.com/sites/SafaricomBloom/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FSafaricomBloom%2FShared%20Documents%2FBloom%20Collections%20%2D%20Inhouse%2FMPESA%20Statements&viewid=aff5ca06%2D3e55%2D414b%2D9b83%2De92efb25265f). The data is loaded into the data warehouse by the “extract\_load\_raw\_mpesa\_transactions” task of the [ETL\_bloom2\_dimensions](https://airflow.asantefsg.com/data-pipelines/dags/ETL_bloom2_dimensions/grid?dag_run_id=scheduled__2023-08-30T23%3A00%3A00%2B00%3A00&task_id=extract_load_raw_mpesa_transactions) DAG.

The mpesa\_statements\_bloom\_data\_recon DAG is scheduled to run every two hours

#### **Configuration Parameters.**

This DAG does not take in any configuration parameters.

#### **Tasks**

1. **share\_disbursements**

This task fetches TEP disbursements data from bloomlive.raw\_mpesa\_transactions table of the data warehouse and loads it into transactions\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task iterates over the fetched data and checks on bloom\_staging.transactions\_data\_dump table whether the records exist.

For records that do not exist in the target database, the task standardizes the records and loads them into the bloom\_staging.transactions\_data\_dump table

1. **share\_repayments**

This task fetches dump repayments data from bloomlive.raw\_mpesa\_transactions table of the data warehouse and loads it into transactions\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task fetches the repayments corresponding disbursement receipt numbers from the data warehouse and merges with the data.

The task iterates over the fetched data and checks on bloom\_staging.transactions\_data\_dump table whether the records exist.

For records that do not exist in the target database, The data is further cleaned and standardized to fit into the bloom\_staging.transactions\_data\_dump table.

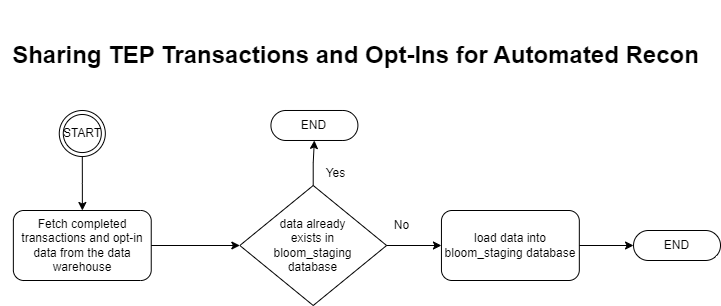
The data is then loaded into the bloom\_staging.transactions\_data\_dump table

1. **trigger\_ETL\_safaricom\_recons\_mifos\_posting**

This task triggers the [ETL\_safaricom\_recons\_mifos\_posting](https://airflow.asantefsg.com/data-pipelines/dags/ETL_safaricom_recons_mifos_posting/grid) DAG

### [[DAG ID: tep\_data\_bloom\_recon](https://airflow.asantefsg.com/data-pipelines/dags/data_dumps_bloom_data_recon/grid)](https://airflow.asantefsg.com/data-pipelines/dags/tep_data_bloom_recon/grid)

#### **Overview**



This data pipeline is designed to load TEP data that is not already posted on MIFOSX into bloom\_staging database. The DAG is triggered by the ETL\_TEP\_dataDAG**.**

#### **Configuration Parameters.**

This DAG does not take in any configuration parameters.

#### **Tasks**

1. **recon\_missing\_opt\_ins**

This task fetches TEP opt-in data from bloomlive.raw\_tep\_client\_activity table of the data warehouse and loads it into opt\_ins\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task iterates over the fetched data and checks on bloom\_staging.opt\_ins\_data\_dump table whether the records exist.

For records that do not exist in the target database, the task standardizes the data and loads them into the bloom\_staging.opt\_ins\_data\_dump

1. **recon\_missing\_tep\_disbursements**

This task fetches TEP disbursements data from bloomlive.raw\_tep\_disbursements table of the data warehouse and loads it into transactions\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task iterates over the fetched data and checks on bloom\_staging.transactions\_data\_dump table whether the records exist.

For records that do not exist in the target database, the task standardizes the records and loads them into the bloom\_staging.transactions\_data\_dump

1. **recon\_missing\_TEP\_repayments**

This task fetches dump repayments data from bloomlive.raw\_tep\_repayments table of the data warehouse and loads it into transactions\_data\_dump table of the bloom\_staging database hosted on the server afsg-dgt-prod-bloom-EC2(10.176.0.53).

The task iterates over the fetched data and checks on bloom\_staging.raw\_tep\_repayments table whether the records exist.

For records that do not exist in the target database, the task fetches the corresponding disbursements receipt numbers from the data warehouse and merges with the data.

The data is further cleaned and standardized to fit into the bloom\_staging.transactions\_data\_dump table schema.

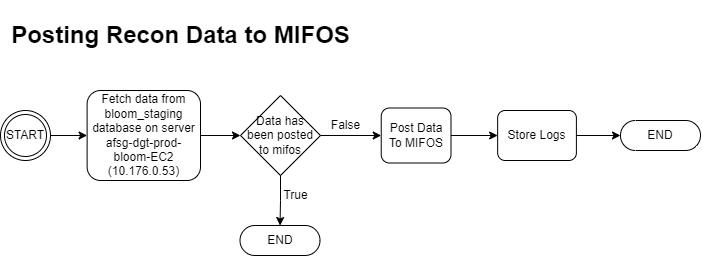
The data is then loaded into the bloom\_staging.transactions\_data\_dump table

1. **trigger\_ETL\_safaricom\_recons\_mifos\_posting**

This task triggers the [ETL\_safaricom\_recons\_mifos\_posting](https://airflow.asantefsg.com/data-pipelines/dags/ETL_safaricom_recons_mifos_posting/grid) DAG

### [[DAG ID: ETL\_safaricom\_recons\_mifos\_posting](https://airflow.asantefsg.com/data-pipelines/dags/data_dumps_bloom_data_recon/grid)](https://airflow.asantefsg.com/data-pipelines/dags/ETL_safaricom_recons_mifos_posting/grid)

#### **Overview**



This data pipeline is designed to post Safaricom Bloom opt-ins, repayments and disbursements reconciliation data from the bloom\_staging database to MIFOSX.

The data pipeline is triggered remotely by the tep\_data\_bloom\_recon, mpesa\_statements\_bloom\_data\_recon and data\_dumps\_bloom\_data\_recon DAGs but it can also be triggered manually.

#### **Configuration Parameters.**

This DAG does not take in any configuration parameters.

#### **Tasks**

**1. post\_opt\_ins**

This task fetches all records in the bloom\_staging.opt\_ins\_data\_dump hosted on server afsg-dgt-prod-bloom-EC2 where the status\_code is null, the store\_number is not ‘-1’ and the national\_id is not ‘None’

The task then proceeds to post the data to MIFOSX via the endpoint url <http://3.123.63.62/bloomAutomation/bloomapi/optIn.php>. The endpoint url is hardcoded in the task

**2. post\_disbursements**

This task fetches all disbursement records in the bloom\_staging.transactions\_data\_dump hosted on server afsg-dgt-prod-bloom-EC2 where the status\_code is null and the store\_number is not ‘-1’.

The task then proceeds to post the data to MIFOSX via the endpoint url <http://3.123.63.62/bloomAutomation/bloomapi/loanRequest.php>. The endpoint url is hardcoded in the task

**3. post\_repayments**

This task fetches all repayment records in the bloom\_staging.transactions\_data\_dump hosted on server afsg-dgt-prod-bloom-EC2 where the status\_code is null, the store\_number is not ‘-1’ and the national\_id is not ‘None’

The task then proceeds to post the data to MIFOSX via the endpoint url <http://3.123.63.62/bloomAutomation/bloomapi/loanRepayment.php>. The endpoint url is hardcoded in the task

**4. send\_ms\_teams\_notification**

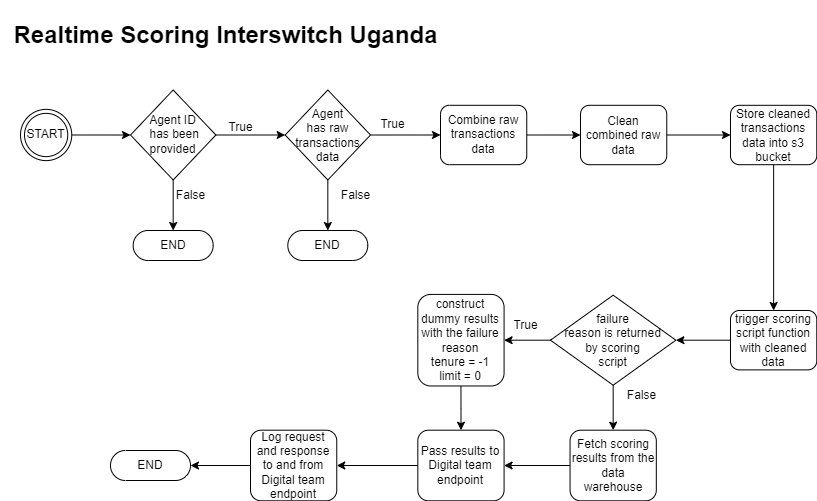
This task shares an accurate summary of the data posted successfully from the bloom\_staging database spanning over the past 7 days.

The notification is shared to the MS Teams channel ‘[Data Dumps](https://teams.microsoft.com/l/channel/19%3aFgrlUsVN31elHVGrtrlnK5ueY3SGtxgRGzhKa0a9IlI1%40thread.tacv2/General?groupId=342cd566-e5f3-49b5-aba5-5ca466260a6a&tenantId=66677b8e-7a4b-4220-b6c5-83da5e50f67c)’

## Product: Interswitch Uganda

### [DAG ID: scoring\_pipeline\_interswitch\_uganda](https://airflow.asantefsg.com/data-pipelines/dags/scoring_pipeline_interswitch_uganda/grid)

#### **Overview**



This data pipeline is designed to score Interswitch Uganda clients. The data pipeline is triggered by Digital team via the [API — Airflow Documentation (apache.org)](https://airflow.apache.org/docs/apache-airflow/stable/security/api.html)

#### **Configuration Parameters.**

The data pipeline expects the “agent\_id”. This is the unique identifier of the client that is to be scored.

Example

{"agent\_id": "3is00362"}

#### **Tasks**

1. **trigger\_scoring**

This task executes the scoring script “interswitch\_uganda.py” located in the “[scoring\_scripts](https://github.com/Asante-FSG/DataPipelines/blob/master/scoring_scripts/remita_nigeria.py)” folder of the DataPipelines project.

To start with, the task checks whether the parameter “agent\_id” has been provided. If not provided, the task is skipped.

If the agent\_id has been provided, the task proceeds to fetch raw

The task calls the function get\_scoring\_results of the interswitch\_uganda.py scoring script and passes cleaned customer data to it.

The raw transactions data for a client is shared by the digital team in the s3 bucket [afsg-dgt-prod-iswug-file-upload](http://afsg-dgt-prod-iswug-file-upload) prior to triggering the scoring DAG. A customer can have more than one file shared for scoring. The files have the agent\_id as the first characters of the name followed by an underscore. For example, file 3IS00106\_23\_06\_23\_07\_22\_7.csv is for the agent 3IS00106

This task fetches the all the files associated with the provided agent id using the “get\_objects” function of the utils.aws\_api module. If files for the agent do not exist in the s3 bucket, the task is skipped.

The task then cleans and combines the data into one dataset. If the dataset is an empty dataset, I.e, the files shared were empty, the message 'All files for {agent\_id} are empty' is logged on Airflow and the task fails with the exception pd.errors.EmptyDataError.

The dataset is uploaded to the s3 bucket [afsg-ds-prod-postgresql-dwh-archive with the naming format {agent\_id}\_cleaned.csv. For example, for agent](https://s3.console.aws.amazon.com/s3/buckets/afsg-ds-prod-postgresql-dwh-archive?region=eu-central-1&prefix=interswitch_uganda/scoring_data/&showversions=false) 3IS00106 the cleaned file would be 3IS00106\_cleaned.csv

After uploading the data to the s3 bucket, the get\_scoring\_results function is called with the cleaned dataset as the value of the “raw\_data” parameter.

The task retrieves the failure\_reason returned by the get\_scoring\_results function and saves it into an [xcom](https://www.bing.com/ck/a?!&&p=418df983c178dda9JmltdHM9MTY5MzI2NzIwMCZpZ3VpZD0yMWMxZjZkNS0zYTk5LTZjN2YtMjNiYy1lNDFlM2IxYjZkMzQmaW5zaWQ9NTE4Mg&ptn=3&hsh=3&fclid=21c1f6d5-3a99-6c7f-23bc-e41e3b1b6d34&psq=airflow+xcom&u=a1aHR0cHM6Ly9haXJmbG93LmFwYWNoZS5vcmcvZG9jcy9hcGFjaGUtYWlyZmxvdy9zdGFibGUvY29yZS1jb25jZXB0cy94Y29tcy5odG1s&ntb=1) variable for subsequent task to use.

1. **pass\_generated\_limits\_to\_engineering**

This task passes the scoring results to the Digital API endpoint.

The task retrieves the “failure\_reason” stored in xcom.

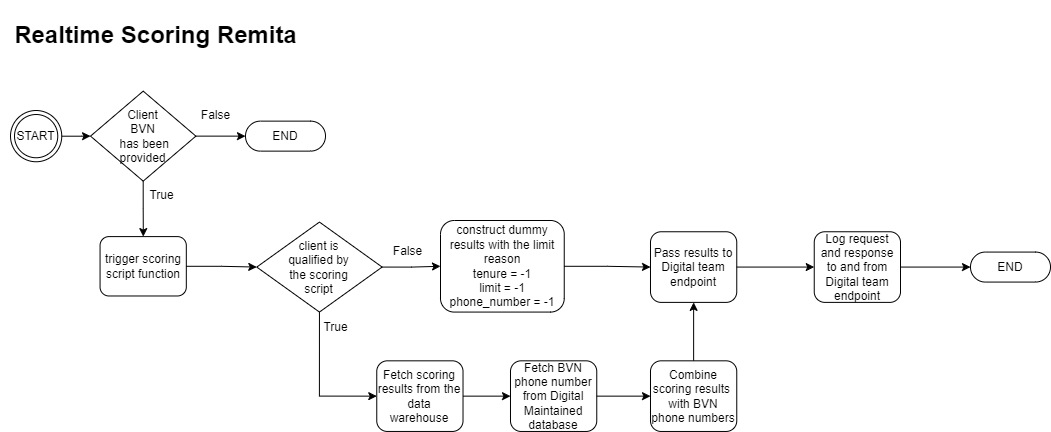
If the failure reason is not None, the task constructs dummy results with the failure reason, tenure set as ‘–1’ and limit set to ‘0’. If the failure reason is None, the task fetches scoring results from the scoring\_results\_view of the interswitch\_ug schema of the data warehouse.

The results, whether fetched from the data warehouse or dummy data, are passed back to Digital team. The raw request and response to and from Digital team are logged onto Airflow.

## Product: Remita

### [DAG ID: scoring\_pipeline\_remita](https://airflow.asantefsg.com/data-pipelines/dags/scoring_pipeline_remita/grid)

#### **Overview**



This data pipeline is designed to score Remita clients. The data pipeline is triggered by Digital team via the [API — Airflow Documentation (apache.org)](https://airflow.apache.org/docs/apache-airflow/stable/security/api.html)

#### **Configuration Parameters.**

The data pipeline expects the two parameters, “bvn” and ”callback\_url”. The bvn is the unique identifier of the client that is to be scored. The callback\_url is the enpoint url to send the scoring results to for Digital team to receive them

Example

{"bvn": 22273266088, "callback\_url": "https://backend.asantefinancegroup.com/remitalive/api/v1/kyc/dsnotification"}

#### **Tasks**

1. **trigger\_scoring**

This task executes the scoring script “remita\_nigerial.py” located in the “[scoring\_scripts](https://github.com/Asante-FSG/DataPipelines/blob/master/scoring_scripts/remita_nigeria.py#L52)” folder of the DataPipelines project.

To start with, the task checks whether the parameter “bvn” has been provided. If not provided, the task is skipped.

If the bvn exists, the task calls the function, get\_scoring\_results of the interswitch\_uganda.py scoring script and passes the bvn as the value of the client\_bvn parameter.

From the scoring script, the task expects a dictionary to be returned, containing the limit reason and a boolean for the final\_is\_qualified key, that indicates whether the customer has been successfully scored or not.

{'limit\_reason': '', 'final\_is\_qualified': False}

The task stores the dictionary into an [xcom](https://www.bing.com/ck/a?!&&p=418df983c178dda9JmltdHM9MTY5MzI2NzIwMCZpZ3VpZD0yMWMxZjZkNS0zYTk5LTZjN2YtMjNiYy1lNDFlM2IxYjZkMzQmaW5zaWQ9NTE4Mg&ptn=3&hsh=3&fclid=21c1f6d5-3a99-6c7f-23bc-e41e3b1b6d34&psq=airflow+xcom&u=a1aHR0cHM6Ly9haXJmbG93LmFwYWNoZS5vcmcvZG9jcy9hcGFjaGUtYWlyZmxvdy9zdGFibGUvY29yZS1jb25jZXB0cy94Y29tcy5odG1s&ntb=1) variable for subsequent task to use.

**2. pass\_generated\_limits\_to\_engineering**

This task sends the scoring results to the url provided in the “callback\_url” DAG configuration parameter.

The task retrieves the dictionary shared by the scoring script from xcom variables and checks the final\_is\_qualified key. If the value is True, the task fetches the scoring results from the scoring\_results\_remita\_view view of the remita schema of the data warehouse.

The task then fetches the mobile number of the client from the Customers table of the remita\_staging database hosted on Digital maintained server 10.0.2.153.

The task adds the mobile number to the scoring results fetched from the data warehouse and shares the results to the callback url.

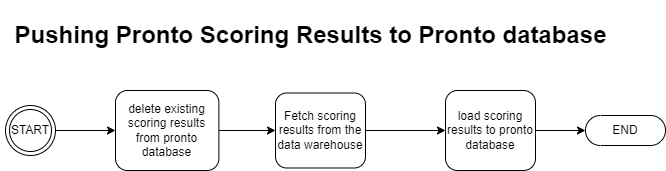
If the value of the final\_is\_qualified key of the dictionary returned by the scoring script is False, the task constructs dummy results with the phone number being –1, limit as –1 and tenure as –1.

Regardless of whether the scoring results are fetched from the data warehouse or dummy data, the limit reason returned by the scoring script is the one passed to the callback url.

The raw request and response to and from Digital team are logged onto Airflow.

## Product: Pronto

### [DAG ID: ETL\_pronto\_scoring\_results](https://airflow.asantefsg.com/data-pipelines/dags/ETL_pronto_scoring_results/grid)



This data pipeline is designed to load scoring results from the data warehouse into Pronto database hosted on server 10.0.2.214. The data pipeline is triggered by the data scientist via the [API — Airflow Documentation (apache.org)](https://airflow.apache.org/docs/apache-airflow/stable/security/api.html)

#### **Configuration Parameters.**

The data pipeline expects the parameter “tasks\_to\_run”. This is the name of the actual task to run in the DAG, that specifies which scoring results to push to Pronto database.

Example

{"tasks\_to\_run": ["load\_rwanda\_ac\_group\_scoring\_results"]}

## [DAG ID: Reports](https://airflow.asantefsg.com/data-pipelines/dags/Reports/grid)

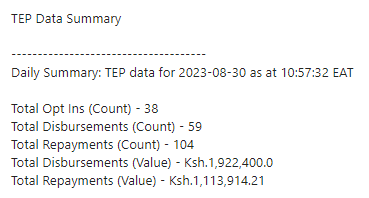
This data pipeline is designed to share reports with Asante FSG team members. It is scheduled to run every 1 hour.

#### **Tasks**

1. **TEP\_data\_report**

This task shares an email report for summary of TEP data to the distribution group[bloom2.0\_migration@asantefinancegroup.com](mailto:bloom2.0_migration@asantefinancegroup.com)

Below is an example of the email sent



The report is consumed by the Operations team and Data Science team. It helps in monitoring TEP data at a near real time basis.

If the email stops being sent, it can be an indicator that the TEP Endpoint maintained by Digital team is down or that the VPN connection between Safaricom and Asante FSG is not working as expected.

1. **closed\_on\_safaricom\_open\_on\_mifos**

This task shares two reports containing all loans that are closed on Safaricom but are still open on MIFOSX.

The first report identifies the loans by selecting only those loans in the loans\_fact\_table\_materialized\_summary\_view of the bloomlive schema that have a principal\_outstanding value that is greater than zero and the safaricom loan balance is 0. The report is a csv file named “cleared\_on\_safaricom\_open\_on\_mifos\_full\_list.csv” and is shared in the sharepoint site '[Data Dumps](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&isAscending=false&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035&view=7&q=cleared%5Fon%5Fsafaricom%5Fopen%5Fon%5Fmifos%5Ffull%5Flist%2Ecsv)'

The second report identifies the loans by selecting all loans that have a contract id value that is in the daily\_closed\_contracts\_dump table of the bloomlive schema and have the loan status as 300. The report is a csv file named “closed\_contracts\_open\_on\_mifos.csv” and shared in the sharepoint site '[Data Dumps](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&sortField=Modified&isAscending=false&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035&view=7&q=closed%5Fcontracts%5Fopen%5Fon%5Fmifos%2Ecsv)'

1. **bloom2\_data\_posting\_summary**

This task shares an email showing the summary of posting of data to MIFOSX.

Below is a sample email



The email is shared to the distribution group [bloom2.0datarecon@netorgft4232141.onmicrosoft.com](mailto:bloom2.0datarecon@netorgft4232141.onmicrosoft.com)

1. **safaricom\_loan\_balances\_summary**

This task shares a report containing safaricom loan balances for each loan bloom2 loan in the bloomlive.loans\_fact\_table\_materialized\_summary\_view

The report is a csv file “safaricom\_balances” and it is shared in the SharePoint site '[Safaricom Bloom](https://netorgft4232141.sharepoint.com/sites/SafaricomBloom)'

1. **bloom\_full\_repayments**

This task shares all records in the bloomlive.repayments\_view of the data warehouse and uploads the csv file “full\_repayments.csv” to the 'Safaricom Bloom' SharePoint site

1. **bloom\_full\_disbursements**

This task shares all records in the bloomlive.repayments\_view of the data warehouse and uploads the csv file “full\_disbursements.csv” to the 'Safaricom Bloom' SharePoint site

1. **bloom\_tep\_repayments**

This task uploads TEP repayments data for the current date from bloomlive.raw\_tep\_repayments table of the data warehouse to the [MS Teams channel](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) ‘Data Dumps’. The file is named “tep\_repayments\_{str(current\_date)}.csv”

1. **solv\_loan\_book\_report**

This task fetches loan data from `mifostenant-tanda`.m\_loan ml table of the core banking system whose products are those in solv\_bat.product\_dimension. The report named “SOLVLOANBOOK{current\_date}.csv” is uploaded to the [SharePoint site 'Data Dumps'](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataDumps%2FShared%20Documents%2FData%20Sharing%2FSOLV%2FSOLV%20LOAN%20BOOK&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035)

1. **bloom\_tep\_disbursements**

This task uploads TEP disbursements data for the current date from bloomlive.raw\_tep\_disbursements table of the data warehouse to the [MS Teams channel](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) ‘Data Dumps’. The file is named “tep\_disbursements\_{str(current\_date)}.csv”

1. **bloom\_tep\_opt\_ins**

This task uploads TEP opt-in data for the current date from bloomlive.raw\_tep\_client\_activity table of the data warehouse to the [MS Teams channel](https://netorgft4232141.sharepoint.com/sites/DataDumps/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FDataDumps%2FShared%20Documents%2FGeneral&viewid=243c7c27%2D009a%2D46e9%2Db208%2D9fb9ac4b2035) ‘Data Dumps’. The file is named “tep\_opt\_ins\_{str(current\_date)}.csv”

## [DAG ID: DVC\_investor\_reports](https://airflow.asantefsg.com/data-pipelines/dags/DVC_investor_reports/grid)

This DAG is designed to trigger the execution of [afsg\_lender\_reports](https://github.com/Asante-FSG/afsg_lender_reports) DVC pipeline. The DAG also uploads reports generated by the DVC pipeline to the SharePoint site [Asante Finance Group - Asante Data Science](https://netorgft4232141.sharepoint.com/sites/AsanteFinanceGroup-AsanteDataScience/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FAsanteFinanceGroup%2DAsanteDataScience%2FShared%20Documents%2FAsante%20Data%20Science%2FLoan%20Tapes%20as%20at%20End%20Months&sortField=LinkFilename&isAscending=true&viewid=ac6b16e2%2D7dd6%2D4992%2D9101%2D3cea53a37e62)

#### **Tasks**

1. **trigger\_dvc\_loan\_tape\_report**

This task makes use of a bash operator to trigger the DVC pipeline.The airflow variable 'DVC\_afsg\_lender\_reports\_project\_path'

1. **upload\_dvc\_loan\_tape\_report\_to\_sharepoint**

This task copies the file loan\_tape\_requests.xlsx from the afsg\_lender\_reports project and uploads it to the SharePoint site [Asante Finance Group - Asante Data Science](https://netorgft4232141.sharepoint.com/sites/AsanteFinanceGroup-AsanteDataScience/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FAsanteFinanceGroup%2DAsanteDataScience%2FShared%20Documents%2FAsante%20Data%20Science%2FLoan%20Tapes%20as%20at%20End%20Months&sortField=LinkFilename&isAscending=true&viewid=ac6b16e2%2D7dd6%2D4992%2D9101%2D3cea53a37e62)