

CNOS GC LA SMO SPLIT

**LLD**

Document Version / Details: Ver. 1.0 / 20-10-2023

WORD DOCUMENT

**VERSION HISTORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **ADO ID** | **Owner** |
| **V1.1** | **17/07/2023** | **NSRD extract generation on 4WD** | **651454** | **Shreyas Mane** |
| **V1.2** | **17/07/2023** | **Updated info on PM exclusion on NIT** | **651446** | **Afnan Hussain** |
| **V.1.3** | **20/10/2023** | **Detailed LLD** |  | **Kshitij Mawale** |

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr no | Contents | Page No |
| 1 | INTRODUCTION | 2 |
| 2 | BUSINESS REQUIREMENTS | 2 |
| 3 | TECHNICAL COMPONENTS | 2 |
| 4 | DATA MODEL DESIGN | 3 |
| 5 | ARCHITECTURE DIAGRAM | 5 |
| 6 | INPUTS | 11 |
| 7 | DATA TRANSFORMATION AND CALCULATION LOGIC | 17 |
| 8 | LA INGESTION | 32 |
| 9 | ADF PIPELINES | 42 |
| 10 | ADB COMPONENTS (TABLES AND VIEWS CREATED) | 76 |
| 11 | AUTOMATED DQ CHECKS | 102 |
| 12 | AAS REFRESH ACTIVITY | 112 |
| 13 | PEA | 113 |
| 14 | DD API / DAS SETUP FOR LA | 121 |
| 15 | MASTER DATA FOR UPLOADER | 133 |
| 16 | DATABASE OBJECTS DIAGRAMS | 135 |
| 17 | DATABASE OBJECTS DEFINITIONS | 135 |
| 18 | LIST OF OBJECTS DEFINITIONS WHICH ARE USED IN AAS | 136 |
| 19 | AAS MODELS AVAILABLE, CONFIGURATION IN AAS MODEL, KPIS CREATION ON FLY IN AAS | 136 |

# INTRODUCTION

* The LA SMO it project aims at improving the existing processing for Latin America CNOS by splitting up the code for each individual SMO, thereby promoting parallel processing and drastically increasing the flexibility of making changes in specific SMOs in the future.
* The LA Region-specific notebooks contain data related to their respective SMO, thereby promoting individuality in the code, reducing the overall runtime, and optimizing pipeline performance.
* Similarly, the Data Factory orchestration is built in a manner that allows for parallel processing, thereby removing dependency between code and reducing overall runtime.

# BUSINESS REQUIREMENTS

* To organize and categorize a collection of input, processing and output notebooks related to the LA region by dividing them based on SMO regions.
* Each table in these replicated notebooks needs to be recreated again for specific SMO.
* Build one single final LA Output table by merging all the SMO specific outputs into one using metadata-based orchestration.
* Implement pipelines in ADF (Azure Data Factory) in such a way that all the SMO’s should run parallel.

# TECHNICAL COMPONENTS

* Azure ADF – To read data and orchestrate complete flow.
* Databricks – To write or apply business logics for CNOS & GC
* Azure Blob – To store data.
* SQL Server – To store dimension and facts
* AAS – To host semantic data models.
* Azure Logic app - To create, develop and deploy cloud-based integrations and workflows.

# DATA MODEL DESIGN

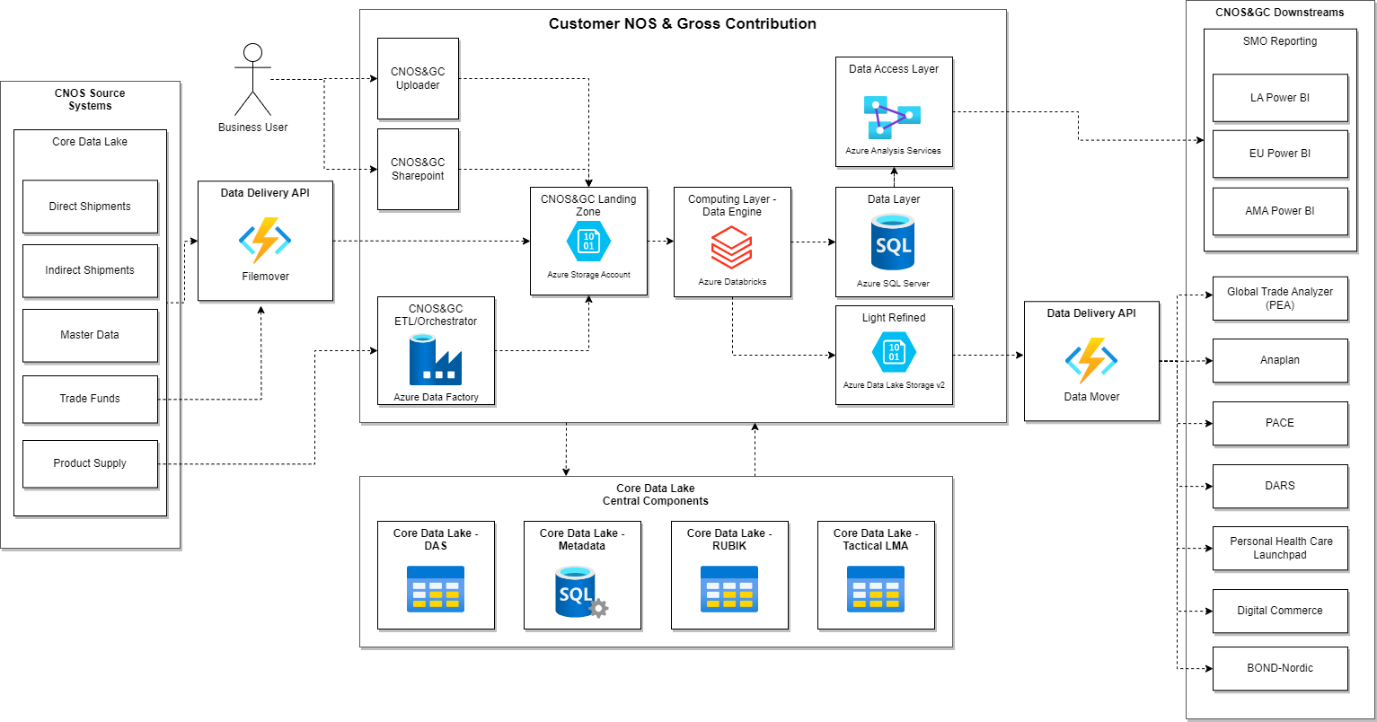
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Measure | Source | Abs./Perc. | Direct/Indirect | Real/Calculated | Comments |
| Volume | Direct Shipments-CDL Refined | Abs. | Direct | Real |  |
| GIV | Direct Shipments- CDL Refined | Abs. | Direct | Real |  |
| NIV | Direct Shipments- CDL Refined | Abs. | Direct | Real |  |
| NIT | Uploader | Abs. | Direct | Real |  |
| NSRD – Damage Returns | CDL PS – SAP Data | Abs. | Direct | Calculated |  |
| NSRD – Price Adjustments | CDL PS – SAP Data | Abs. | Direct | Calculated |  |
| NSRD – Distributor Discounts | CDL PS – SAP Data | Abs. | Direct | Calculated |  |
| NSRD – Prompt Payment | CDL PS – SAP Data | Abs. | Direct | Calculated |  |
| NSRD – Logistics Discounts | CDL PS – SAP Data | Abs | Direct | Calculated |  |
| NSRD – Others | CDL PS – SAP Data | Abs | Direct | Calculated |  |
| NSRD – Write Off | CDL PS – SAP Data | Abs. | Direct | Calculated |  |
| SAP Gross Sales | CDL PS – SAP Data | Abs. | Direct | Calculated |  |
| CAT NSRD Tieout | Calculation | Abs. | Direct | Calculated |  |
| CAT SD Tieout | Calculation | Abs. | Direct | Calculated |  |
| CAT TDC Tieout | Calculation | Abs. | Direct | Calculated |  |
| NSRD Manual Input | CDL PS – SAP Data | Abs | Direct | Calculated |  |
| SD Manual Input | Optima-Uploader | Abs | Direct | Calculated |  |
| NSRD Total | Calculation | Abs. | Direct | Calculated |  |
| SD Total | Calculation | Abs. | Direct | Calculated |  |
| SD – Live Rates | Optima-Uploader | Abs | Direct | Real |  |
| SD – Fixed Budget | Optima- Uploader | Abs | Direct | Real |  |
| GC | Calculation | Abs. | Direct | Calculated |  |
| NOS | Calculation | Abs. | Direct | Calculated |  |

**4.1 INPUT DATA SOURCES:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Input | Input | Manual/Automated | Source | Frequency of Refresh | Target WD |
| **Functional** | **Customer Mapping** | Manual | MDM | Monthly | 4WD |
| **Functional** | **Product Mapping** | Automated | MDM | Monthly | 4WD |
| **Non-Functional** | **Geo Mapping & PC Mapping** | Automated | MDM | Monthly | 4 WD |
| **Functional** | **Volume, GIV, NIV** | Automated | Direct Shipments-CDL Refined | Monthly | 3WD |
| **Functional** | **NIT** | Automated | Uploader | Monthly | 7WD |
| **Functional** | **NSRD, Gross Sales** | Automated | CDL PS – SAP Data | Monthly | 3WD |
| **Functional** | **FMR Tieouts** | Automated | Uploader | Monthly | 7WD |
| **Functional** | **TDC** | Automated | Uploader | Semi Annual | 7WD |
| **Functional** | **SD - Live Rates** | Automated | Optima – Uploader | Monthly | 7WD |
| **Functional** | **SD – Fixed Budget** | Automated | Optima - Uploader | Monthly | 7WD |

# ARCHITECTURE DIAGRAM

## CNOS & GC Architecture Diagram



**5.1.1 CNOS Source Systems**

* The Source System of CNOS and GC is Core Data Lake (CDL). It includes Direct Shipments, Indirect Shipments, Master Data, Trade Funds, Product Supply.
* For Direct and Indirect Shipment, the extracted KPIs in scope are Volume MSU, GIV, NIV. The data is extracted in transactional currencies and converted to local currency and aggregated after the conversion.
* Master Data such as Customer Mapping and Product Mapping is updated in the RUBIK and raw parquet files are available in the CNOS Blob storage.
* Uploader Tool is a web-based application, specifically created for CNOS & GC for maintaining fact data/manual inputs. Input SPOCs work on a downloaded excel file which they will fill with their inputs and upload these inputs to the application.
* Similarly, Business User updates and provides flat files on SharePoint, which is copied into a CNOS & GC Landing Zone BLOB Storage during processing.
* DD API is a P&G-created API that sends various sets of data from CDL Refined to any subscriber/downstream.

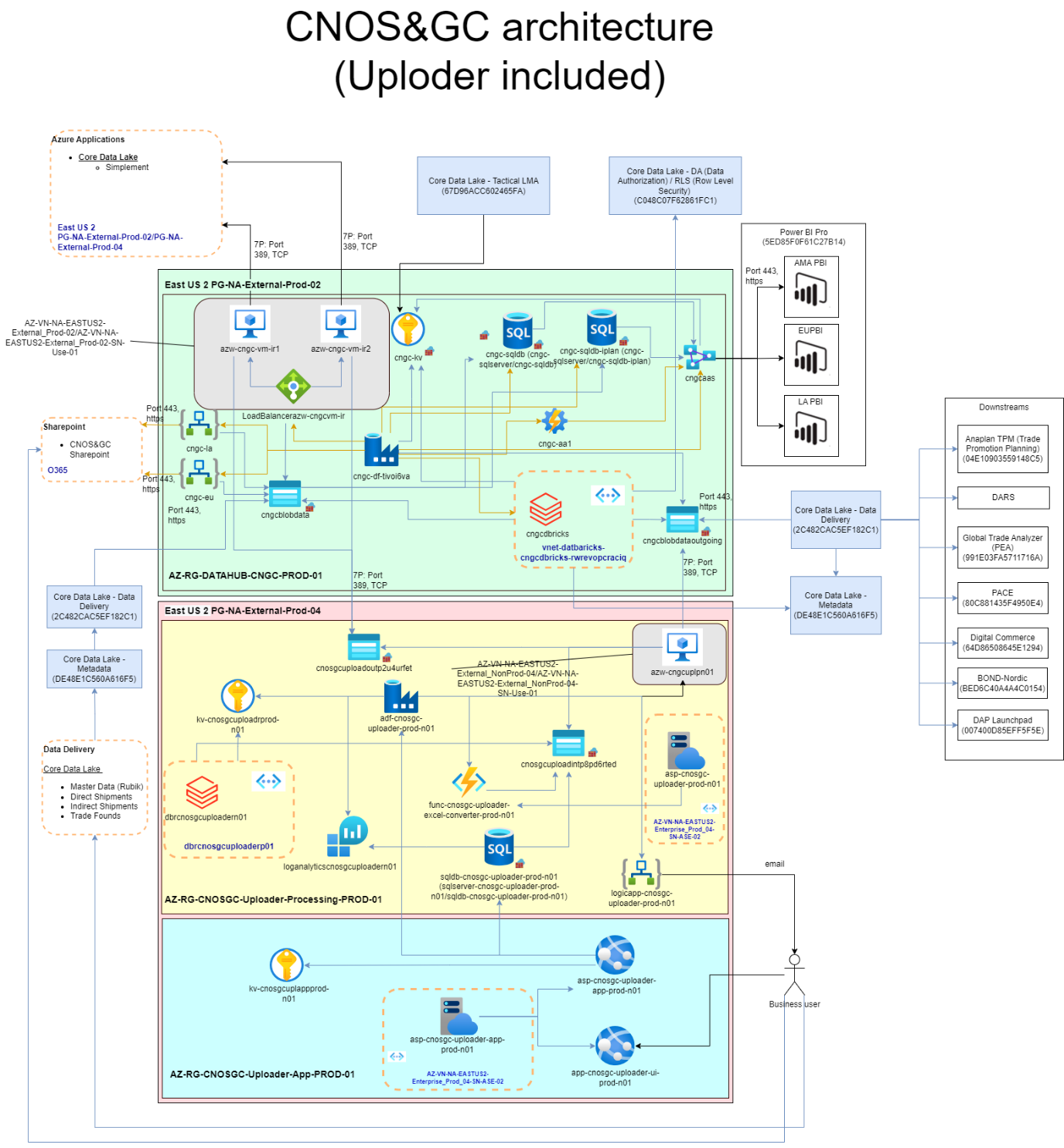
**5.1.2 CNOS Core Processing**

* The CNOS and GC application uses meta data approach to process the data.
* The SQL server contains metadata for workday processing and based on the Workday the Azure Data Factory (ADF) pipelines are triggered automatically to perform related activities.
* The data from Landing Zone (Azure BLOB) is copied to Incoming Zone via Azure Data Factory pipelines and further it is copied to Processing Zone.
* The Processing Zone data is the source for CNOS, which is ingested into Integration layer with Databricks, and respective tables and views are created from the available data.
* The Computing Layer Process the data with Data Transformation and Calculation Logic built in Databricks notebooks to provide output reports to the users.
* Automated Data Quality check framework is created to capture the input and output data discrepancy and stop the processing based on test case conditions.
* Azure Analysis Services (AAS) is used for connecting to the calculated outputs in Azure Databricks and exposing the data via Pivot Table to the users.

**5.1.3 CNOS Downstream**

* CNOS & GC Raw output files are published after a transformation process and shared with approved users either in raw form or via end-user reporting.
* Customer NOS & GC has multiple downstream and shares data in different forms such as AAS, Flat Files and CDL ( Via DD API).
* Notifications regarding the CNGC Raw Output data extractions are shared via email communication (automated via Logic apps orchestration) with approved users.
* GC Output files per country will be shared with the central DARS team for other reporting purposes.
* Power BI dashboard is used for SMO reporting for Europe, Latin America, and AMA region.
* PEA and ANAPLAN outputs are shared via Flat files on SharePoint.
* PHC output is limited to Personal Health Care Subsector, and it is shared downstream via CDL publishing.

## CNOS & GC architecture with Uploader



The Above Diagram can be split in three parts based on the Subscriptions and resource Groups:

1. US EAST 2 - PG-NA-External-Prod-02 - AZ-RG-DATAHUB-CNGC-PROD-01 - CNOS Core Calculation Engine
2. US EAST 2 - PG-NA-External-Prod-04 - AZ-RG-CNOSGC-Uploader-Processing-PROD-01 – Uploader App (Backend)
3. US EAST 2 - PG-NA-External-Prod-04 - az-rg-cnosgc-uploader-app-prod-01 – Uploader Application. (Frontend)

### CNOS Core Calculation Engine

## Core Data Lake - Azure Application (CDL PS)

Used for sourcing of SAP objects like glpca, t001, vbrk, vbrp.

## Core Data Lake (Data Delivery and Metadata)

Used to deliver objects such as Direct Shipments, Indirect Shipments, Corporate Hierarchies, Product / Customer Mapping from Refined Shipments, Rubik respectively to CNOS Layer.

cngc-df-tivoi6va

ADF is the orchestrator of the application, responsible for running ingestion pipelines, running processing / output generation notebooks, running logic apps for email, flat file sharing and refreshing AAS with latest data.

LoadBalancerazw-cngcvm-ir

azw-cngcvm-ir1 and azw-cngcvm-ir2

These are the VM’s which are used as integration runtime for the cngc-df-tivoi6va. An Integration runtime is essential to power the ADF. These VM’s are placed in a load balancer to distribute workload when usage is high.

vnet-datbaricks-cngcdbricks-rwrevopcraciq

cngcdbricks

All the ingestion, processing, DQ check, Output generation logic is written using SparkSQL / Pyspark in Azure databricks. These notebooks are executed via pipelines from cngc-df-tivoi6va. The databricks instance is placed in a Virtual Network (vnet-datbaricks-cngcdbricks-rwrevopcraciq), all the networking configuration is done using this VNET.

## cngc-sqldb (cngc-sqlserver/cngc-sqldb)

The database cngc-sqldb belongs to the server **cngc-sqlserver.** The database contains all the tables / views / stored procedures required for the Orchestration (ADF uses metadata approach). The SQL database also maintains the results of the DQ checks, required for PowerBI report. The Analysis Service also sources its data from the SQL database.

**cngcblobdata** is the internal storage account of CNOS and GC. All the data from External sources (DD-API, CDL PS, Uploader) is copied into this storage account in various containers. All Objects required for processing are sourced from this Storage account. All the intermediate objects generated during processing are also maintained here.

**cngcaas** the processed data is shared with the users using AAS. Row Level Security is also implemented here. AAS sources its data from the SQL database. Multiple KPI’s are calculated in AAS on fly using DAX formulae.

cngc-la (Logic App) and Sharepoint

The Flat file outputs are shared with the users via SharePoint. There exists an Office 365 Connection between the Logic App and SharePoint which enables the Logic App to copy the flat files from the blob container and place it into SharePoint. This activity is controlled via metadata in SQL.

cngc-kv

This is a Key-vault instance, all the secrets (Keys, Service Principals, etc) are maintained here.

cngcblobdataoutgoing

The Output generated as part of the databricks code is saved in this storage accounts. This Account is the source for Blob to SharePoint activity, CDL light refined publishing, some downstream directly connect to the storage container using SAS tokens to access the output files.

## Data Delivery API and Data Authorization Service (DAS)

The output generated in the CNOS Layer, is shared to the downstreams using DD-API.

**cngcblobdataoutgoing** is the sources for the DD-API file mover. DAS provides Row level security configuration for each subset of data.

## PowerBI Pro

PowerBI is used for visualization of data and creation of reports for business use. It uses the data maintained in AAS as its source.

## Tactical LMA

Tactical LMA is configured on a key – vault. They send reminder emails when secrets in a vault are about to expire. Generally, reminders start a month before expiry.

## Uploader Application

The Manual inputs required in the CNOS Layer are maintained and provided in the uploader Application. It is an interface for the users to upload, validate and maintain data.

## Uploader Backend

**sqldb-cnosgc-uploader-prod-n01**

**(sqlserver-cnosgc-uploader-prod-n01/sqldb-cnosgc-uploader-prod-n01)**

Whenever an Upload/Download action is performed, stored procedures are executed via ADF. In case of an upload action, the new data is temporarily copied into a stage table and validated.

Once the validation is completed, it is merged into worktable. This worktable is used to display data on frontend and retrieve data using the download functionality. All the metadata and results of test cases are also maintained here.

**kv-cnosgcuploadrprod-n01**

This is a Key-vault instance, all the secrets (Keys, Service Principals, etc) are maintained here.

**cnosgcuploadintp8pd6rted**

This is the internal storage account of uploader, input from CNOS Layer (required for Load) is copied in this storage account and further processed. All the configuration files maintained in the DB are also copied here during the Loading process. All intermediate tables (Load Cubes / Load Stage Tables) are saved as parquets files in this storage account.

cnosgcuploadoutp2u4urfet

This is the external storage account of uploader. All the input templates (after Upload Action is completed) are saved here as parquet files. This storage account is connected to the CNOS layer, and all the input templates required for GC processing are copied from this storage account.

adf-cnosgc-uploader-prod-n01

The ADF is the Orchestrator of the Application. All the stored procedures / notebooks / Function App / Logic App runs are handled by the ADF.

It is used for running the data loading pipelines. It is also used when an upload or download activity is triggered by the user. Respective ADF pipelines are called via REST API. These pipelines further deal with SQL server for validation and merging of data.

azw-cngcuplpn01

This is a VM’s which is used as integration runtime for the adf-cnosgc-uploader-prod-n01. An Integration runtime is essential to power the ADF.

dbrcnosgcuploaderp01vnet

dbrcnosgcuploaderp01

All the input data (Shipments / Hierarchies / Mapping) is ingested as tables in ADB. Further this data is processed to created Load cubes and Load stage Tables. These Stage tables are then copied to the Database where they are merged with existing worktables to get rows associated with latest month.

The databricks instance is placed in a Virtual Network (dbrcnosgcuploaderp01vnet), all the networking configuration is done using this VNET.

logicapp-cnosgc-uploader-prod-n01

This is a logic app which uses an Office 365 connection to share Emails with the users. These Email are generated on (Upload Completion / Failure). A separate email is generated for the Ops and technical team upon load completion / failure.

asp-cnosgc-uploader-prod-n01

Azure Functions runs in an App Service plan. An App Service plan defines a set of compute resources for a web app to run. This app service plan helps the execution of function App.

func-cnosgc-uploader-excel-converter-prod-n01

Azure Functions is a serverless solution that allows to write less code, maintain less infrastructure, and save on costs. This function app supports Upload and download activity. It has the capability of converting a file from xlsx to csv and csv to excel in the process of upload and download.

## Uploader Frontend

asp-cnosgc-uploader-app-prod-n01

An app service always runs in an App Service plan. An App Service plan defines a set of compute resources for a web app to run. To run app service of UI and API this app service plan is used.

app-cnosgc-uploader-ui-prod-n01

Azure App Service is an HTTP-based service for hosting web applications. This App service is used to host UI component of Uploader. UI supports all the layout which is visible in the Uploader application.

app-cnosgc-uploader-app-prod-n01

Azure App Service is an HTTP-based service for hosting web applications, REST API’s. This App service is used to host API component of Uploader. API is used to interact with UI and Database. Based on request it will fetch the data from DB, process the data and then it will send the response to UI.

kv-cnosgcuplappprod-n01

This is a Key-vault instance, all the secrets (Keys, Service Principals, etc) related to UI and API are maintained here.

# INPUTS

## Volume and GIV and NIV

* Data is extracted automatically from Direct Shipments- CDL Refined.

Customer IDs to be used for data extraction and aggregation will be defined based on Customer Mapping input data from LA Data Hub MMR ShipTo ID with a many-to-one mapping with Optima Account ID. So extracted data can be aggregated and matched to Optima funds.

* Data (GIV)extracted from CDL refined is in transactional currencies, which is assumed to be the local currency. If GIV per geo ID in systems of record is available in multiple transactional currencies; value shipments will be simply aggregated together without translation into 1 local currency. Local currency mapping is maintained in Uploader as part of Geography mapping which is used as a multiplier factor to convert transactional currencies into local currencies.
* Local currency code is mapped based on geo levels from Geo mapping table as geo Id to create currency geo ID table.
* Extracted KPI’s in scope: GIV, Volume, NIV
* Customer\_IDs are extracted from the MDM CM and customer manual mapping.
* Currency code is converted from ‘$’ to Local currency code using FX rates.

The following filters are added to extract LA specific region data:

* Profit Center
* GEO\_ID
* GEO NAME
* ORG\_ID! =’9999’
* prod\_csu\_type\_code<>'D'
* CORP\_OFFCL\_SHIP\_FLAG='Y'
* GIV or NIV or Profit Center or GEO\_ID is NOT NULL
* Time Period as PY and Current FY

## CUSTOMER MAPPING

**Customer Mapping Preperation Steps:**

1. Customer Mapping from MDM is joined with SAP & Shipments ingested data to categorize the Cust L9 IDs based on measures the Customer is associated with. This data includes attributes like, Reporting customer name, customer name, geography, regional channel, regional subchannel etc.

2. Different edge case and possible actions (Based on the Description Column)

1.1 **HUB**- These are HUB only Ship to IDs, no corresponding match found either in Infopage or in SAP. These can be eliminated from the base customer mapping.

1.2 **SAP\_HUB**- These are SAP account ID matching the HUB but not found in Infopage Shipments. Evita to manually map to the Ship to IDs so that there is no potential loss of Gross Sales and NSRD if found for these SAP Account IDs. Map them to the Ship to ID which are either SAP\_HUB\_SHIP or SHIP\_HUB type of Description.

1.3 **SAP**- These are SAP Account ID Only and not found in the Hub. LA RSMO/SMO should map them to the Ship to ID which are either SAP\_HUB\_SHIP or SHIP\_HUB type of Description.

1.4 **HUB\_SHIP**- These are the records present in both HUB and shipments. Customer which are present in HUB and having either Vol or GIV associated with it. We should still consider this for Customer Mapping Base. However, we will not receive Gross Sales and NSRD for these Customers.

1.5 **SHIP, SAP\_SHIP**- This use case should not exist at all.

1.6 **SAP\_HUB\_SHIP**- These are the full match of customer in all the systems and can be considered for the customer mapping base.

**Final Customer Mapping – Customer L9s with description as: SAP\_HUB\_SHIP & SHIP\_HUB**.

**This process is part of following notebooks:**

CNGC\_LA\_SMO/<SMO\_NAME>/20\_PREPROCESSING\_VIEWS\_<SMO\_NAME>/206\_SIMPLEMENT\_VIEWS\_<SMO\_NAME>

CNGC\_LA\_SMO/<SMO\_NAME>/20\_PREPROCESSING\_VIEWS\_<SMO\_NAME>/

400\_CREATE\_LADMAR\_INPUTS\_TC\_TO\_LC

CNGC\_LA\_SMO/<SMO\_NAME>/20\_PREPROCESSING\_VIEWS\_<SMO\_NAME>/110\_CREATE\_CSV\_TO\_SP\_<SMO\_NAME>

## FX Rates Sourcing Model – MMR & CDL refined

First CDL data is pulled from table stage\_dir\_shipments.masterdata\_exchg\_rate\_fct and then Joined with table stage\_mmr.mnr\_frgn\_exchg\_rate\_dim\_vw\_v which is sourcing fx rates from uploader table stage\_uploader.MNR\_FRGN\_EXCHG\_RATE\_DIM\_MV Union result of above join with USD-to-USD converstion from stage\_mmr.mnr\_frgn\_exchg\_rate\_dim\_vw\_v.

Data coming from stage\_mmr.mnr\_frgn\_exchg\_rate\_dim\_vw\_v has priority over data coming from stage\_dir\_shipments.masterdata\_exchg\_rate\_fct – Uploader fx rates will be used first and cdl fx rates will be used when mmr fx rates are not available.

For Argentina fx rates are source from Uploader where a separate table is created in common layer stage\_common.fx\_rates\_argentina.

Which is later union with stage\_common.exchg\_rate\_fct\_mmr\_cdl\_merge and new table  stage\_common.frgn\_exchg\_rate\_dim\_v is created which consist of fx rates for all other SMO’s?

**This process is part of following notebooks:**

/Shared/CNGC\_COMMON/30\_PREPROCESSING\_COMMON\_DIMS/300\_CREATE\_COMMON\_DIMS

Object Created

stage\_common.exchg\_rate\_fct\_mmr\_cdl\_merge

source Table used:

stage\_mmr.exchg\_rate\_fct\_01

stage\_mmr.mnr\_frgn\_exchg\_rate\_dim\_vw\_v

## PRODUCT MAPPING

* LA SMO owns Product mapping maintenance in MDM, and GC engine extracts the Product mapping from LA Data hub and prepares final version of Product Mapping.
* Product mapping from MDM is maintained and ingested at Brand level, following the 5005-product hierarchy.
* Product mapping is segregated into two separate sets of code, one for Japan, and the other for all non-Japan SMOs.
* Standard model of Product Mapping Automatic update: Product hierarchy is generated from RDS, along with Geo Dimension. LA Data User to mark the products which are for EXCLUSION (Indicating these products are to be excluded from GC Reporting)

## PROFIT CENTER AND GEO MAPPING

* The purpose of the Geo Mapping Technical Template is to provide full scope of Geo Dimension data that will be used to further create MDM input templates for Customer Mapping, Product Mapping and Coefficient data. Geo Mapping together with Profit Center & Market Mapping will be the sole source of Geo, Profit Center, and Product dimension information for all further data processing. Geo mapping is sourced from MDM.
* The purpose of Profit Center & Market Mapping Technical Template in MDM is to set up the filters used for Shipment Data Pull and for further CNGC Data Processing and to allow for dynamic changes in filtering of the shipment data if needed. The template will be updated on a yearly basis at the beginning of the new FY. The updates will be maintained by the CNGC Team based on business requirements.

## NIT

Data is extracted automatically from Uploader.

**The following filters are added to extract LA specific region data:**

* Source Level Extraction Details - From SAP (L6P)
* Transaction ZGNIT\_AUDIT or table ‘zgxx\_nit\_audit’ & ‘New Table Name’
* Period, ZG\_BSR\_PERIOD = MM.YYYY for July FY18/19 type 01.2019
* Process Execution, ZG\_NIT\_PEXEC\_CNT = 1
* Execution Type, ZG\_NIT\_PEXEC\_TYPE = 1
* Company Code, BUKRS = 2378
* RFC Destination, RFCDEST = FSSERS-G4P470-L6P
* Layout, SLIS\_VARI = /NITAUDIT
* GEO\_ID(geo id filter consist of smo wise value)eg. for Argentina geo\_id=’032’,for Mexico geo\_id=’484’ etc.
* orig\_le ='501'
* proft\_ctr\_3\_id=’09437’
* CUSTM\_SMO\_NAME
* NIT\_AMT is NOT NULL

**This process is part of following notebooks:**

/Shared/CNGC\_LA\_SMO/01\_ARGENTINA/20\_PREPROCESSING\_VIEWS\_ARGENTINA/205\_RLS\_OPTIMA\_NIT\_CM\_VIEWS\_<SMO\_NAME>

Object created:

VIEW stage\_la\_excels.manual\_nit\_v\_<SMO\_NAME>

Source Blob/View/Table Read in Step:

stage\_uploader.la\_nit

INNER JOIN stage\_uploader.MNR\_LE\_MAPNG\_DIM\_MV

left join stage\_common.sode\_g707\_fdim.

inner join stage\_common.create\_prod\_hier\_join\_filters\_PROD\_5005

inner join  stage\_product\_mapping\_exc\_jpn\_mdm.prod\_map\_geo\_mapping\_final

## NSRD

**NSRD are extracted from Simplement-GLPCA**

Source Level Extraction Details: SAP L6P - Source

**The filters applied to GLPCA tables are:**

* CUSTM\_SMO\_NAME
* proft\_ctr\_3\_id =’09437’
* simp\_chng\_type\_code <> 'D'
* RBUKRS(rbukrs filter consist of smo wise value) for eg, rbukrs=’501’ for Argentina,’682’ for Brazil etc
* RRCTY = '0' AND RVERS = '000' AND KOKRS = 'XX00'
* Table - GLPCA, and below filters applicable ·
* Record Type, RRCTY = 0 ·
* Version, RVERS = 0 ·
* Posting Period, POPER = FY Month (Example – 1 for July 2 for August)
* Company Code, RBUKRS = 2378, .... (Based on Country we are fetching the data for; mapping is mentioned in NSRd Mapping Reference data document)
* Controlling Area, KOKRS = XX00
* Account Number, RACCT = 30300000 to 30319999 (Account ID to NSRd type mapping is mentioned in NSRd Mapping Reference data document)
* Fiscal Year, REFRYEAR = 2020 (Example 19/20: 2020)

**This process is part of following notebooks:**

/Shared/CNGC\_LA\_SMO/01\_ARGENTINA/10\_INGESTION\_ARGENTINA/107\_SIMPLEMENT\_DATA\_<SMO\_NAME>

Object Created:

* TABLE stage\_la\_simplement.glpca\_cdl\_<SMO\_NAME>\_BASE

Source Blob/View/Table Read in Step:

* path='dbfs:/mnt/cngc-storage/processing/simplement\_cdl/GLPCA'

## NSRd Manual Input

* Source:Uploder
* Template Name: Customer Manual Input
* Technical view name of the template: CUST\_MANUL\_INPUT
* Template to be created in Uploaderr for Data Input SPOC to load
* NSRd Manual Inputs.
* NSRd Manual Inputs value is provided in LC.
* Data is provided at Reporting Customer - Category level – Month – FY
* Allocate to FPC level in final output based on % GIV.

**This process is part of following notebooks:**

/Shared/CNGC\_COMMON/10\_INGESTION/103\_MMR\_INGESTION

Shared/CNGC\_COMMON/20\_PREPROCESSING\_VIEWS/202\_MMR\_VIEWS

## NSRd FMR

* Source: Uploader
* Template Name: FMR Inputs
* Technical view name of the template: FM\_RPTNG\_INPUT
* NSRd FMR is provided in USD, will need to be converted to LC for base calculations. (Mentioned as part of calculation section for Tieouts)
* Template created in Master Minder for Data Input SPOC to load NSRd FMR
* Data is provided at upon GEO – Country level – Month – FY

**This process is part of following notebooks:**

/Shared/CNGC\_COMMON/10\_INGESTION/103\_MMR\_INGESTION

/Shared/CNGC\_COMMON/20\_PREPROCESSING\_VIEWS/202\_MMR\_VIEWS

## SD LIVE RATES

* SD LIVE RATES is extracted from Uploader.
* Input is taken from tables stage\_uploader.sd\_live\_rates \_la\_<SMO\_NAME>
* Values are provided at Reporting Customer – Product 5005 Hierarchy Level
* Columns to consider – Account ID, Fund ID, Product Level, Product ID, Transaction Date & Actual Accrued Fund.

**This process is part of following notebooks:**

/Shared/CNGC\_LA\_SMO/02\_<SMO\_NAME>/20\_PREPROCESSING\_VIEWS\_\_<SMO\_NAME>/205\_RLS\_OPTIMA\_NIT\_CM\_VIEWS\_<SMO\_NAME>

## SD FIXED BUDGET

* SD FIXED BUDGET is extracted from Uploader.
* Input is taken from tables stage\_uploader.sd\_fixed\_budget \_la\_<SMO\_NAME>
* Values are provided at Reporting Customer – Fund Name Level
* Columns to consider – Account ID, Fund Name, Transaction Date & Actual Accrued Fund.

**This process is part of following notebooks:**

* /Shared/CNGC\_LA\_SMO/02\_<SMO\_NAME>/20\_PREPROCESSING\_VIEWS\_\_<SMO\_NAME>/205\_RLS\_OPTIMA\_NIT\_CM\_VIEWS\_<SMO\_NAME>

## FMR TIEOUTS

* FMR TIEOUTS are extracted from Uploader.
* Input is taken from ‘mnr\_fm\_rptng\_input\_dim\_vw\_v’ view.
* Values are provided in USD, to use Periodic FX rates for conversion to LC.

The following filters are applied to extract LA specific region data:

* CUSTM\_SMO\_NAME
* GEO\_ID
* PROD\_HIER\_ID = 5005
* NSRD\_TIE\_OUT\_LC or SD\_TIE\_OUT\_LC or TDC\_TIE\_OUT\_LC is NOT NULL

**This process is part of following notebooks:**

* /Shared/CNGC\_COMMON/20\_PREPROCESSING\_VIEWS/202\_MMR\_VIEWS

# DATA TRANSFORMATION AND CALCULATION LOGIC

## OPTIMA Feed Logic

**Description:**

**SD Live rates** (Base ship funded) are provided at Reporting Customer & various Product Levels.

At present, we are getting SD Live Rates at Sub Sector, Category and Brand levels. Now, we may also get SD Live Rates at Company Level and Sector level, which are Level 2 and 3 of the PRODUCT Hierarchy 5005. (Company level having Product ID as ‘1000013908’)

Product Levels against which we get SD Live Rates is

Company – Level 2

Sector – Level 3

Sub sector – Level 4

Category – Level 5

Brand – Level 6

**The allocation finally is to happen at FPC level.**

Steps were carried out earlier in Notebook **650\_LA\_SD\_DIM\_FACT\_PROCESS\_<SMO\_NAME>** for Subsector, Category and Brand levels. with addition to that, SD live rates at company and subsector allocation is also part of same notebook mentioned above.

**Steps:**

1. Valid FUND IDs from the Optima Base Ship funded files are to be considered. Valid FUND IDs are maintained in stage\_uploader.MNR\_VAL\_FUND\_OPT\_DIM\_MV table which is created by using Uploader Template ‘Valid FUND IDs Optima’.
2. Once the SD funds are filtered based on Valid FUND IDs list. Allocation must take place.
3. The values provided are at Reporting Customer – Product Dimension – Month level.
4. Allocate SD Live rates to respective Customer L9s, mapped to Reporting Customer (This mapping will come from manual\_customer\_mapping file provided)
5. Split the allocated based on % GIV
6. Process is the same even if we get the SD funds at Company & Sector level.
7. Check the shipments of Customer L9s
8. Allocate the SD FUNDs based on % GIV for all Cust L9s having shipments

## NSRd Design

1. During allocation of NSRD If a record doesn’t specify a Customer ID for a particular row, then, exclude it from allocation step.
2. Customer ID included is the Customer Level 9 (Corporate Customer Hierarchy 898)
3. If Profit Center = “1DDEFAULT” and has a customer code, by using GIV% for the given customer.
4. If Profit Center = Any other value other than 1DDEFAULT and has material number linked – Then allocated NSRd to FPC Level
5. If Profit Center = Any other value other than 1DDEFAULT and has NO material number linked - work on a customer / profit center basis and allocate down to FPC. (**PC – Subsector mapping is maintained in Uploader**)
6. Identification of NSRd type is based on the account number. Refer to column A & column C in the document ‘NSRd Mapping Reference Data’ (**Mapping template will be configured in Uploader**)
7. The Profit Center in NSRd extract is based upon 084 Corporate Profit Center Hierarchy (Level 7). Level 3 of the same hierarchy matches the CNGC standard 064 PC hierarchy. GC Engine will fetch 084 hierarchy from RDS and figures Level 3 PC node of the mentioned L7 084 PC node in the extract.
8. If Account Number is not found in Key for “NSRd Type” use “Other” and raise issue with R-SMO SPOC to include for next cycle
9. Posting Date provides info about which month the NSRd data corresponds to.
10. Template in Uploader for Account IDs & NSRd type – Updated Half-yearly.

A diagram of a diagram

Description automatically generated

### 6.2.1 Approach:

If a given customer is falling under more than one possible scenarios, then we should apply the stated logic for both independently and sum up the value for a given customer and assign.

All the NSRd amount are to be accounted for.

It’s like we perform allocation based on scenario 1 > Take the updated data as base and perform assign/allocations for values from scenario 2 > Take scenario 2 values as base and assign/allocate values we get from Scenario 3.

### 6.2.2 NSRd Type & Account ID Mapping sample:

|  |  |
| --- | --- |
| 30310001 | Returns & Adjustments |
| 30310002 | Pricing Adjustments |
| 30310003 | Distributor Discounts -SD |
| 30310004 | Distributor Discounts -Manual |
| 30310005 | Returns & Adjustments - Finished Products |
| 30310006 | Returns & Adjustments for Export Sales |
| 30310007 | Other Deductions from Sales-Other (SD) |
| 30310008 | Other Deductions from Sales-Sales-Another (COPA)-manual |
| 30310009 | Other Deductions from Sales-Other (SD) SA00 |
| 30310010 | Cash Discount - Manual |
| 30310011 | Cash Discount Granted -Auto (AR) |
| 30310013 | Streamlined Lgst Glbl Base Plan Disc I |
| 30318004 | Bad Debts - Unresolved Customer Deductions - COPA |

**Reference Mapping Documents to be used for NSRd:**

1. Account ID- NSRd Type mapping: This provides information of which Account ID corresponds to which NSRd type
2. PC – Subsector Mapping: This mapping provides data about which subsectors are valid for a given PC (084 hierarchy). This is used when we have only NSRd value at only Customer & PC (084 hierarchy) level and no FPC provided.
3. Legal Entity – Country Mapping: This mapping provides info about which Legal Entity from source corresponds to which Country. Maintained as part of Geo mapping in MMR.

|  |  |  |
| --- | --- | --- |
| **Receiver Code** | **Legal Entity Country** | **SMO** |
| 501 | Argentina | Argentina |
| 682 | Brazil | Brazil |
| 536 | Chile | Chile |
| 295 | Colombia | Colombia |
| 398 | Peru | Peru |
| 2378 | Mexico | Mexico |
| 656 | Costa Rica | LADMAR |
| 718 | Guatemala | LADMAR |
| 585 | Ecuador | LADMAR |
| 396 | Panama | LADMAR |
| 819 | Panama | LADMAR |
| 2379 | USA Special Customers (PSMT) | LADMAR |

Legal Entity & NSRd type are maintained as part of Reference mapping templates in Uploader

## Gross Sales & NSRD Allocation design flow

1. NO Sales scenario allocation logic – Design mentioned as per document attached below
2. Pre-Walmart allocation design – Customers with NO Shipments but having GS/NSRD values are to be modified. The way to manipulate this data set having GS/NSRD values is by transferring values to other customers which possess Shipments & mapped to same reporting customer.
3. Walmart Scenario allocation design - Aggregate the derived GS values to Reported Customer Level & allocate it down to Cust L9s based on % GIV. The same approach is to be followed for NSRds (From SAP) as the source table for both Gross Sales & NSRd is the same.

### 6.3.1 Walmart Scenario

Allocation of Gross sales at Cust L9 levels, making them look in sync with GIV.



### 6.3.2 Pre-Walmart Scenario

Reallocation of GS / NSRD values from customers who do not have Shipments to those who do, for the same Reporting Customer / TIME\_ID / GEO\_ID / CURRENCY\_ID.

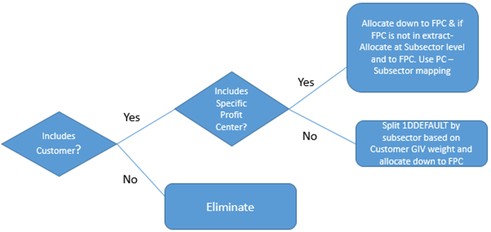
********

**Notebook used for the allocation Process:**

Shared/CNGC\_LA\_SMO/01\_<SMO\_NAME> /40\_PROCESSING\_REPORTING\_<SMO\_NAME>/750\_LA\_NSRD\_GS\_FACT\_PROCESS\_<SMO\_NAME>

## Gross Sales Design

* 1. During allocation If a record doesn’t specify a Customer ID for a particular row, then, exclude it from allocation step
  2. Customer ID included is the Customer Level 9 (Corporate Customer Hierarchy 898)
  3. If Profit Center = “1DDEFAULT” and has a customer code, split amount to subsector by using GIV% for the given customer.
  4. If Profit Center = Any other value other than 1DDEFAULT and has material number linked – Then allocated Gross Sales to FPC Level
  5. If Profit Center = Any other value other than 1DDEFAULT and has NO material number linked - work on a customer / profit center basis and allocate down to FPC
  6. Identification of Gross Sales (In comparison to NSRd) is based on the account number.
  7. The Profit Center mentioned is based upon Corporate Profit Center Hierarchy ‘084’. GC Engine to fetch the 084 L3 PC of the mentioned PC in the extract and map it against Base or Export PC.
  8. Gross Sales– Account Type mapping is to be maintained in Uploader
  9. Posting Date provides info about which month the NSRd data corresponds to.



### 6.4.1 Approach

* + If a given customer appears under more than one possible scenario, then we should apply the stated logic for both independently and sum up the value for a given customer and assign.
  + All the NSRd amounts are to be accounted for.
  + It’s like we perform allocation based on scenario 1 > Take the updated data as base and perform assign/allocations for values from scenario 2 > Take scenario 2 values as base and assign/allocate values we get from Scenario 3.
  + Gross Sales – Account ID Type Uploader Template –

|  |  |
| --- | --- |
| **Account** | **Type** |
| 30000100 | Gross Sales |
| 30000001 | Gross Sales |
| 30008001 | Gross Sales |
| 30008886 | Gross Sales |
| 30000015 | Gross Sales |
| 30000500 | Gross Sales |

Legal Entity – Country Mapping: This provides info about which Legal Entity from source corresponds to which Country. Maintained as part of Geo mapping in Uploader.

|  |  |  |
| --- | --- | --- |
| **Receiver Code** | **Legal Entity Country** | **SMO** |
| 501 | Argentina | Argentina |
| 682 | Brazil | Brazil |
| 536 | Chile | Chile |
| 295 | Colombia | Colombia |
| 398 | Peru | Peru |
| 2378 | Mexico | Mexico |
| 656 | Costa Rica | LADMAR |
| 718 | Guatemala | LADMAR |
| 585 | Ecuador | LADMAR |
| 396 | Panama | LADMAR |
| 819 | Panama | LADMAR |
| 2379 | USA Special Customers (PSMT) | LADMAR |

Notebook used for the allocation Process:

Shared/CNGC\_LA\_SMO/01\_<SMO\_NAME> /40\_PROCESSING\_REPORTING\_<SMO\_NAME>/750\_LA\_NSRD\_GS\_FACT\_PROCESS\_<SMO\_NAME>

## NIT Design

A screenshot of a table

Description automatically generated

* + Data from SAP will be at Customer – FPC Level, and in Local Currency.
  + NIT value will be subtraction of two columns – OS in LC & SDED CO LE. Mockup in screenshot above.

Legal Entity – Country Mapping: This provides info about which Legal Entity from source corresponds to which Country. Maintained as part of Geo mapping in MMR.

|  |  |  |
| --- | --- | --- |
| **Receiver Code** | **Legal Entity Country** | **SMO** |
| 501 | Argentina | Argentina |
| 682 | Brazil | Brazil |
| 536 | Chile | Chile |
| 295 | Colombia | Colombia |
| 398 | Peru | Peru |
| 2378 | Mexico | Mexico |
| 656 | Costa Rica | LADMAR |
| 718 | Guatemala | LADMAR |
| 585 | Ecuador | LADMAR |
| 396 | Panama | LADMAR |
| 819 | Panama | LADMAR |
| 2379 | USA Special Customers (PSMT) | LADMAR |

Notebook used for the allocation Process:

Shared/CNGC\_LA\_SMO/01\_<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/600\_LA\_NIT\_FACT\_PROCESS\_<SMO\_NAME>

## NSRd Logistics Discount

1. NSRd logistics discount values are to be sourced & ingested from 2 sources (SAP & Optima SD Live Rate file)
2. Extraction from SAP and allocation logic is already explained in NSRd logics section (Which is determination of NSRd logistics discount based on account ID– PC mapping and then going ahead with allocations).
3. Extraction from SD Live Rate (Base Ship FUNDED) Optima dump is based on Fund IDs. From the file sourced from Optima.
4. Filter out NSRd logistics discount based on Fund IDs
5. Values available at Customer – Product 5005 hierarchy level
6. Then Allocate NSRD Logistic Discount, down to FPC level, based on % GIV basis.
7. These will be part of the same template where SD Valid FUND IDs are maintained. Filer by – KPI Measure = NSRd Logistic Discount

A screenshot of a computer

Description automatically generated

On High level:

1. First complete the SAP Allocations
2. Then perform Optima allocations (SDs & NSRd logistic discount).

## SD TPR

SD TPR value is to be made available at Customer – FPC Level, for which two sets of extracts are to be merged.

1. Take SD TPR extract which is maintained in MMR at Billing Document – Product (FPC) level
2. Fetch VBRK extract from CDL, which is mapping between Billing Document - Payer (Customer) Combine both these to get SD TPR at Customer – FPC Level. Allocation happens at FPC level directly after this.

* SD TPR is maintained in Uploader on Monthly basis
* Every Month, data is archived, and template is made empty for data input spoc to make the upload.
* A screenshot of a computer

  Description automatically generated

A table with numbers on it

Description automatically generated

Scenarios Observed:

* Same FPC – Customer to have similar/duplicates rows for same period. This is a valid scenario.
* A screenshot of a data

  Description automatically generated

Azure to read the values in same way as it did with manual extract. We perform aggregation at Customer – FPC level from Input and assign the KPI value.

NIT value of Customer (2002903060) – FPC (80298077) for Feb’19 is 941.57 + 941.57 = 1883.14

**Notebook used for the allocation Process:**

Shared/CNGC\_LA\_SMO/01\_<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/650\_LA\_SD\_DIM\_FACT\_PROCESS\_ARGENTINA

## TDC Completeness and Accuracy

TDC Values allocation for FPCs which are part of Input file –

* Consider only the TDC EDS file based on the SMO. If Base PC, then select the Input file from EDS. And if Export PC, then select the Input file which is from MMR.
* Once the TDC input for the FPC is identified – Then convert that value to LC using the TDC specific FX rate. And the converted value will be TDC/SU in LC.

If more than one row for a given FPC, then go for Max (TDC) value.

The first step is to fill out missing FPC level TDC data based on upper level TDC/SU average. All weighted average is based on Country – Category – Brand dimension, it does not include customer specific weight. The process is the same for current and past year TDC/SU processing.

1. Remove all products where TDC/SU and Volume equals to 0
2. If volume <> 0 and TDC/SU = 0, then apply weighted averages of Product Hierarchy 5005 to FPC.
3. Weighted Averages preferences is from Lower Product Hierarchy – Brand> Category
4. It’s like if Brand form weighted Average equals 0 then apply Brand weighted average, if Brand weighted average equals 0, then apply category weighted average TDC/SU to FPC.

Need to add transformation from FPC to CUGTIN – including logic where multiple CUGTINs are mapped to one FPC.

## Customer Manual Inputs Calculation logic (SD & NSRD Manual Inputs)

* Data is provided at Reporting Customer - Category level – Month – FY
* Allocate to FPC level in final output based on % GIV.

There is also the possibility that a CMI is provided for Reporting Customer – Category – Month level which doesn’t possess shipments for that month. As allocation down to FPC is difficult in this case with no GIV, No Sales Scenario also applies to CMI allocation.

Customer Manual Inputs No Sales

Resolution to “no sales scenario” – LA CUSTOMER MANUAL INPUT



## TIEOUTs Calculation Logic

KPIs involved in calculation of Tieouts – FMR Inputs, all other inputs related to NSRd & SD.

As FMR Inputs are provided at Category level in Uploader, which is sourced from SELECT \* FROM fm\_rptng\_input table the Tieouts are calculated at Category level.

1. FMR Inputs provided in $ rate is converted to LC using periodic FX Rate
2. All other KPIs are aggregated at category level
3. Formula is then processed at Category level
4. The value obtained for a given category is aggregated down to FPC level, based on %GIV value of that FPC for that given category.
5. If, give category doesn’t have shipments for that month, then Tie Outs is calculated using the shipments pattern (CFY & PFY months) of that specific category.

## Currency Translations

1. For all Input KPIs & Base Calculations (TieOuts, Totals, GC & NOS) use Periodic FX Rate for LC to C$ & R$ conversions and vice versa. (**Note** – For TDC FMR also Periodic Fx rate is to be used)
2. For TDC (EDS) measure which is provided in $. Below steps are followed for TDC specific Currency conversions –
   1. TDC (From EDS) is provided in $ - From source system.
   2. $ value from above step is converted to LC using TDC specific June FF / Dec FF rates
   3. After we convert to LC, then periodic fx rates is used

## TDC $ to FX conversion

Always latest TDC fx rate to be used for Previous FY & Current FYTD.

FYTD. Example:

1. Let’s say we are in July month (FY1920) or October month (FY1920)

|  |  |  |
| --- | --- | --- |
| FY | June FF | Dec FF |
| FY1819 | 1 | 1.2 |
| FY1920 | 1.3 |  |

We use June FF rate as conversion factor ($ to LC) for all the reporting months (PY & Current FY)

1. Let’s say we are in January month (FY1920) or February month (FY1920)

|  |  |  |
| --- | --- | --- |
| FY | June FF | Dec FF |
| FY1819 | 1 | 1.2 |
| FY1920 | 1.3 | 1.25 |

We use Dec FF rate as conversion factor ($ to LC) for all the reporting months (PY & Current FY)

## CALCULATIONS

Totals are calculated using the formulae at the lowest level possible using formulae.

|  |  |  |  |
| --- | --- | --- | --- |
| **Measure** | **LC** | **R$** | **C$** |
| GROSS SALES | Gross Sales\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_PROMPT PAYMENT | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_LOGISTIC DISCOUNTS | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_DAMAGE & RETURNS | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_DISTRIBUTOR DISCOUNTS | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_PRICE ADJUSTMENTS | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure  x FY YTD PL ordinary average FX rate |
| NSRd\_WRITE OFF | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_OTHERS | NSRd\_LC, Filter by Account ID | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_MANUAL\_INPUT | NSRd\_Manual Input LC (MMR) | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NIT | NIT\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate rate |
| Required NSRd from SAP (Theoretical) | Sum of NSRd breakdowns (Which is NSRd\_LC in below formulas) – NIT |  |  |
| NSRd\_TIE OUT | (NSRDd\_FMR X 1/ FX Rate)-  NSRd\_LC – NIT LC -NSRd Manual Inputs LC) | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NSRd\_TOTAL | NSRd LC + NIT LC + NSRd Manual Input LC + NSRd Tie Out LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| SD\_TPR | SD\_TPR\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| SD\_LIVE RATES | SD\_LIVE RATES\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| SD\_FIXED BUDGET | SD\_FIXED BUDGET\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| SD\_MANUAL INPUT | SD\_MANUAL INPUT\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| SD\_TIE OUT | (SD(FMR) x 1/ FX Rate) - SD\_TPR\_LC - SD\_LIVE RATES\_LC  - SD\_FIXED BUDGET\_LC - SD\_MANUAL INPUT\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| SD\_TOTAL | SD\_TPR\_LC + SD\_LIVE RATES\_LC  + SD\_FIXED BUDGET\_LC + SD\_MANUAL INPUT\_LC + SD TIE OUT\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| NOS | Gross Sales\_LC – NSRdTotal\_LC - SD Total\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| TDC | TDC/SU USD x Volume x 1/TDC FX Rate | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| TDC\_TIE OUT | (TDC (FMR) x 1/Periodic FX Rate – TDC\_LC) | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |
| GC | NOS\_LC – TDC\_LC - TDC Tieout\_LC | = LC Value of the measure x FX rate last closed month FYTD | = LC Value of the measure x FY YTD PL ordinary average FX rate |

## TieOuts NoSales scenario

TIEOUTs Calculation Logic and Technical implementation of No Sales scenario for TieOuts.



## NSRD Prompt Payments Exclusion for Mexico PHC Products

Adjust the existing allocation logic for Mexico NSRd Prompt Payments via hard coding.



# LA INGESTION

This section includes information about DD API Subscription & flow to Storage, Shipments &MD refinement after sourcing, Uploader Ingestion and CDL PS Connection Type / Integration methodology of SAP data.

## Ingestion of Direct Shipments and MDM Objects (not including Customer / Product Mapping).

**Pipeline:** 62\_COMMON\_Orchestration/common\_3wd-pipeline

**Frequency of Ingestion:** Monthly

DD – API Subscription Details:

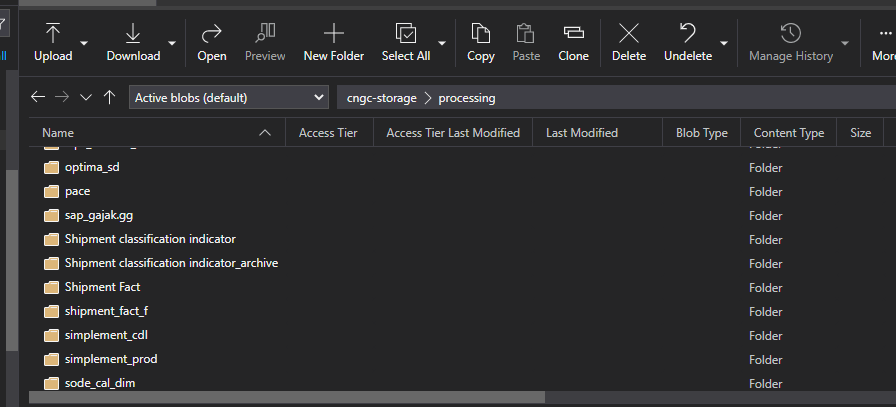
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object Name** | **Target Container** | **Delivery Type** | **Trigger Type** | **Publisher** | **Scope** |
| refined\_direct\_shipments | landing-zone-dir-sh | Incremental | Event based | Direct Shipments | Global |
| cust\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| cust\_hier\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| cust\_dim\_ext\_generic | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| prod\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| prod\_hier\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| prod\_dim\_ext\_generic | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| org\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| org\_hier\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| profit\_center\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| profit\_center\_hier | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| currency\_lkp | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| geo\_hier\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| prod\_life\_cycle\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| uom\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| prod\_bom | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| prod\_uom\_assoc | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| exchange\_rate\_lkp | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| exchange\_rate\_type\_lkp | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
| shipments\_classification\_ind | landing-zone-dir-sh | Full Scope | Event based | Direct Shipments | Global |
| cal\_dim | landing-zone-mdm | Full Scope | Event based | Rubik | Global |
|  |  |  |  |  |  |

All the data is copied from its respective target container to cngc-storage/incoming folder and from incoming layer the data is copied to the cngc-storage/processing layer using the inner activities of 3WD common pipeline.

**Direct Shipments**

A screenshot of a computer

Description automatically generated



1. The First Pipeline Copies data till the incoming layer from the landing-zone
2. The Second pipeline copies data from incoming layer to processing layer

Like Direct Shipments,

**MDM Objects** also copy data in two Phases viz:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. The First Pipeline Copies data till the incoming layer from the landing-zone
2. The Second pipeline copies data from incoming layer to processing layer (cngc-storage/processing/mdm\_dim)

## Ingestion of Shipments Objects in ADB

Once the data is copied into the Processing folder, the Ingestion of Direct Shipments into ADB is carried out by following notebook.

/Shared/CNGC\_COMMON/10\_INGESTION/106\_DIRECT\_SHIPMENTS\_REFINED

The Data for Direct Shipments is maintained as 2 separate folders in the Storage viz. Full Scope (one-time full Load), Incremental data (Copied every month).

This notebook executes in the following way.

1. CMD 1- Performs ingestion of Shipments Classification Indicator.
2. CMD 2 –Performs ingestion of Full Scope of Direct Shipments
3. CMD 3 – Performs ingestion of Incremental data for Direct shipments
4. CMD 4 – Performs union tables created in step 2 & 3 to create a single object for Direct Shipments

## Ingestion of MDM Object in ADB

After completion of copy to the processing layer the objects are ingested into ADB using the following notebook.

/Shared/CNGC\_COMMON/10\_INGESTION/107\_MDM\_Dimensions

The MDM objects are delivered to storage account in a way that, each iteration generates a new Process Run Key in the storage container. The approach is to select the latest (maximum) Process Run Key because the data is complete and not incremental. Including all Process Run Keys would lead to duplication.

Once the Extra Process run keys are deleted the Latest (max) key is ingested in the ADB layer.

This process is uniform for all the MDM related objects.

## Ingestion of MDM objects (Customer and Product Mapping Only)

**Pipeline:** 57\_MDM\_ingestion\_processing/pipeline\_LA\_MDM\_ingestion

**Frequency**: Monthly

A separate pipeline (pipeline\_LA\_MDM\_ingestion) is used to ingest the customer and product mapping from the MDM. This pipeline is triggered from **cngc\_la\_ingestion\_archiving.**

Object for customer mapping – cust\_dim\_ext\_generic

Object for product mapping – prod\_dim\_ext\_generic

The following files with other required files (prod\_dim, prod\_hier\_dim, cust\_dim, cust\_hier\_dim) are copied to a separate folder in the processing layer (processing/mdm).

A screenshot of a computer

Description automatically generated

The first step in the pipeline is to clean up the landing zone, this is done to remove the extra process run keys in the source container itself (to reduce copy time). Further, the data is copied in two phases landing-zone to incoming layer and incoming layer to processing layer.

Furthermore, **customer mapping** is processed in the notebook:

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/104\_LA\_CUSTOMER\_MAP

The possessing steps are as follows:

1. Create Tables for Cust\_dim, cust\_hier\_dim, Cust\_dim\_ext\_generic
2. Filter the cust\_hier\_dim for hier\_id = 898
3. Join cust\_dim with geo\_hier = 707
4. Join the table created in step (c) with cust\_hier\_dim on cust\_key
5. Filter the table created in step (d) using filter curr\_ind = ‘Y’ and active\_ind = ‘Y’
6. Join the table created in step(e) with cust\_dim\_ext\_generic on cust\_key

Filter the final table with following data\_provider\_codes 'ADP\_LA\_LA',

    'ADP\_LA\_ARGENTINA',

    'ADP\_LA\_BRAZIL',

    'ADP\_LA\_CHILE',

    'ADP\_LA\_COLOMBIA',

    'ADP\_LA\_LADMAR',

    'ADP\_LA\_MEXICO',

    'ADP\_LA\_PERU'

(These are the data Providers relevant to LA)

1. Alias the columns as per their general naming convention used in the further part of the code.

For Product Mapping, the MDM objects are processed in the following notebook.

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/105\_PRODUCT\_MAP\_EXC\_JPN

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/112\_PRODUCT\_MAP\_EXC\_JPN\_Adhoc

The following are the steps followed during the notebook execution.

1. Create Tables for prod\_dim, prod\_hier\_dim, prod\_dim\_ext\_generic
2. Filter the prod\_hier\_dim for hier\_id = 5005
3. Join the table created in step (b) with prod\_dim
4. Filter the table created in step (c) using filter curr\_ind = ‘Y’ and active\_ind = ‘Y’
5. Join the table created in step(d) with Prod\_dim\_ext\_generic on prod\_key
6. Filter the table created in step (e) using the following filters

'ADP\_LA\_ARGENTINA',

   'ADP\_LA\_BRAZIL',

   'ADP\_LA\_CHILE',

   'ADP\_LA\_COLOMBIA',

   'ADP\_LA\_LADMAR',

   'ADP\_LA\_MEXICO',

   'ADP\_LA\_PERU'

These are the data providers relevant to LA.

1. Alias the columns as per their general naming convention used in the further part of the code.

## Uploader Ingestion

**Pipeline:** 55\_uploader/pipeline\_cp\_and\_process\_uploader\_data

**Frequency**: Daily

The uploader files are copied into the CNGC Layer from the uploader Outgoing Storage Account (cngcbp2/input\_files)

This storage account is linked to Data Factory via SAS token.

The Copy, following the general guidelines happens in two steps

A diagram of a pipeline

Description automatically generated

1. Uploader Storage Account (cngcbp2) to incoming.
2. Incoming to processing.

Once the data is copied to the processing layer, the following four notebooks are called via data factory.

A screenshot of a computer

Description automatically generated

1.Uploader\_data\_categorization:

/Shared/CNGC\_COMMON/10\_INGESTION/UPLOADER\_PREPROCESSING/100\_UPLOADER\_CATEGORIZATION

1. The Uploader templates received as parquet files have uploaders internal naming convention (example: input\_001\_0234\_work\_fct)

Location: cngc-storage/processing/uploader\_source\_data/

1. Using this name, it is not possible to identify a file and allocate a category to it. Hence the Identification and categorization of the files happens based on subset of columns present in the file.
2. Categories of the files are declared in the code and each category consists of certain column or combination columns of which if a subset is present in the file, the file is assigned that category.

Example:

"LA\_FIXED\_PERU": [

        "TRANSACTION\_ID",

        "FIXED\_PERU",

    ]

This means a category LA\_FIXED\_PERU must consist of a columns TRANSACTION\_ID and FIXED\_PERU.

Any file containing columns: TRANSACTION\_ID and FIXED\_PERU will be categorized as LA\_FIXED\_PERU and all such file will be dumped in a single folder named LA\_FIXED\_PERU.

1. Similarly, categorization happens in a loop for each input template and each template will be assigned one category based on which column combination is the subset of column list.
2. If no columns match, other category is assigned to the input template.
3. Later old files are cleaned up and replaced with newly categorized files.
4. Uploader\_ingestion

/Shared/CNGC\_COMMON/10\_INGESTION/UPLOADER\_PREPROCESSING/110\_UPLOADER\_INGESTION

1. All the categorized files are ingested into ADB using this notebook.
2. All the data which is read is in parquet format.
3. Filters are applied to the data, if required by business Logic.
4. Uploader\_archivization

/Shared/CNGC\_COMMON/10\_INGESTION/UPLOADER\_PREPROCESSING/120\_UPLOADER\_ARCHIVE

1. Uploader Ingestion on daily basis, with every ingestion the old files are archived to cngc-storage/uploader\_archive/
2. Old Input templates are archived in a folder with a timestamp.

## CDL PS Connection Type / Integration methodology of SAP data

**Pipeline Name:** cngc\_la\_ingestion\_archiving

Frequency: Monthly

The following 4 datasets are maintained in SAP:-

* t001
* vbrk
* vbrp
* glpca

This data is extracted from SAP via CDL PS (Simplement CDL) and dumped as parquets in a storage container.

A screenshot of a computer

Description automatically generated

From here the data is copied to the cngc – layer in two steps

A screenshot of a computer

Description automatically generated

1. CDL PS storage container to cngc-incoming
2. cngc-Incoming to cngc-Processing
3. The Extraction from SAP to Parquet is managed entirely by CDL PS.
4. Ingestion into ADB is done as part of the /Shared/CNGC\_LA/10\_INGESTION/ 107\_SIMPLEMENT\_DATA\_<SMO> Notebook.
5. The above notebook is executed as part of each SMO execution.
6. Simplement Data extraction is utilized for NSRD calculations, Currency mapping, Customer mapping.
7. The table ‘glpca\_cdl\_SOURCE\_<SMO\_NAME>’ is created by reading data from files located at 'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/GLPCA' and a new table is created called glpca\_cdl\_<SMO\_NAME> based on the previous table, but only selecting data where ‘rbukrs’ column equals ‘<SMO\_SPECIFIC\_VALUE>’.
8. The table vbrp\_cdl\_SOURCE\_<SMO\_NAME> is created by reading data from files located at 'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/vbrp/' and a new table is created called vbrp\_cdl\_<SMO\_NAME> based on the previous table, but only selecting data where ‘lland\_auft’ column equals ‘SMO\_SPECIFIC\_VALUE’.
9. The table ‘t001\_cdl\_SOURCE\_<SMO\_NAME>’ is created by reading data from files located at 'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/ t001/' and a new table is created called ‘t001\_cdl\_SOURCE\_<SMO\_NAME>’ based on the previous table, but only selecting data where ‘bukrs\_glob’ column equals ‘<SMO\_SPECIFIC\_VALUE>’.
10. The table ‘glpca\_backup\_SOURCE\_<SMO\_NAME>’ is created by reading data from files located at 'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/glpca\_backup' and a new table is created called ‘glpca\_backup\_<SMO\_NAME>’ based on the previous table, but only selecting data where ‘rbukrs’ column equals ‘<SMO\_SPECIFIC\_VALUE>’.
11. These exact same tables have been created for the rest of the SMOs as well, but with the corresponding SMO name and SMO specific filters applied.

# ADF PIPELINES

## Datasets and Linked Services:

List of services and data set is attached here:



## METADATA

* [cngc\_operations].[CNOS\_GC\_ETL\_Notebooks\_source\_LA] table is the main control table where the list of notebooks which are active (Execution Enabled as “Y”) and to be executed are present with details like Execution Day,SMO,Sequence,Subject Area, Notebook Name ,Notebook Path, and Execution Enabled Flag.
* For any newly created notebook to be considered for execution, there should be an entry corresponding to that notebook in [cngc\_operations].[CNOS\_GC\_ETL\_Notebooks\_source\_LA] table with mandatory information for Execution\_Day,SMO,Subject\_Area,Sequence,Notebook\_Name,Notebook\_Path,Execution\_Enabled fields.

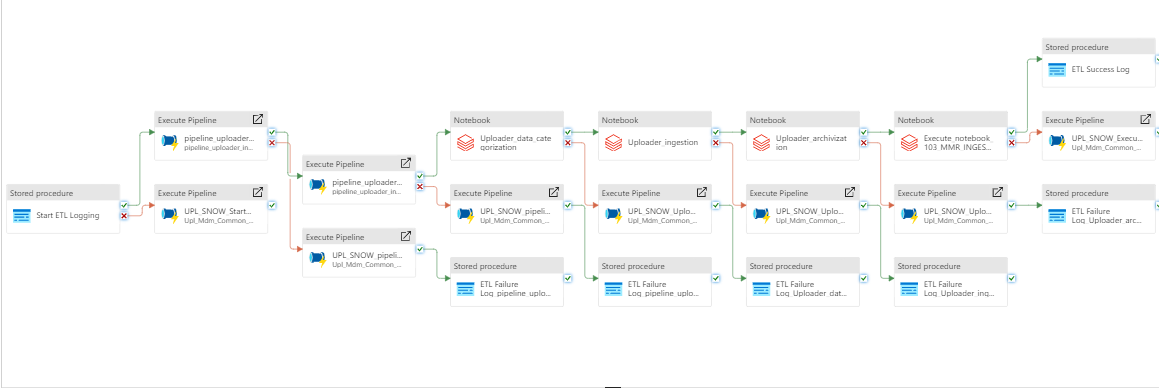
Example:

* [cngc\_operations].[CNOS\_GC\_ETL\_Notebooks\_source\_LA] ( [Execution\_Day], [SMO], [Subject\_Area], [Sequence], [Notebook\_Name], [Notebook\_Path], [Execution\_Enabled], [ServiceNow\_Location], [Last\_Inserted\_Date])
* SELECT ‘7WD','GENERAL','GENERAL','1','100\_CREATE\_SCHEMAS','/Shared/CNGC\_COMMON/10\_INGESTION/100\_CREATE\_SCHEMAS','Y','LATIN AMERICA',getdate().

## LIST OF PIPELINES

## 8.3.1 Uploader pipelines:

## 8.3.1.1 pipeline\_cp\_and\_process\_uploader\_data



Purpose:

This pipeline is used to ingest the uploader data and run the uploader categorization code and finally create the relevant tables from user provided data in uploader.

Important activities of pipelines are explained below:

1. Copy data from uploader layer to CNGC-incoming folder

A screenshot of a computer

Description automatically generated

1. copy data from CNGC-incoming folder to Processing folder

A screenshot of a computer

Description automatically generated

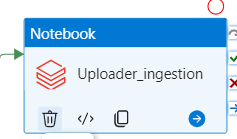
1. run notebook: - /Shared/CNGC\_COMMON/10\_INGESTION/UPLOADER\_PREPROCESSING/100\_UPLOADER\_CATEGORIZATION

A screenshot of a computer

Description automatically generated

1. run uploader ingestion notebook:

/Shared/CNGC\_COMMON/10\_INGESTION/UPLOADER\_PREPROCESSING/110\_UPLOADER\_INGESTION



1. run notebook for archive:

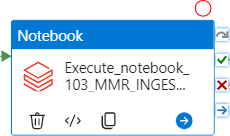
/Shared/CNGC\_COMMON/10\_INGESTION/UPLOADER\_PREPROCESSING/120\_UPLOADER\_ARCHIVE

A screenshot of a computer

Description automatically generated

1. run Notebook:

/Shared/CNGC\_COMMON/10\_INGESTION/103\_MMR\_INGESTION



## MDM pipelines:

## 9.3.2.1 pipeline\_LA\_MDM\_ingestion:

A white rectangular box with text

Description automatically generated with medium confidence

Purpose:

This pipeline is used to ingest MDM data for customers and product mapping together.

Inner activities are explained below:

1. Fetch Blob path and copy data to incoming folder:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Lookup will execute the query and fetch the path from SQL server of MDM source

A close-up of a computer code

Description automatically generated

While copy activity copies data to CNGC incoming folder.

1. This activity provides the source and destination to move earlier copied data to processing folder

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

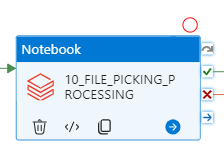
Foreach activity will be irate through each fetched path:

A screenshot of a computer

Description automatically generated

1. Execute notebook:

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/10\_FILE\_PICKING\_PROCESSING



1. Execute notebook:

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/100\_CREATE\_SCHEMAS

A screenshot of a computer

Description automatically generated

1. Execute Notebook:

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/104\_LA\_CUSTOMER\_MAP

This notebook is dedicated for ingesting only LA customer mapping.

A screenshot of a computer

Description automatically generated

1. Execute Notebook:

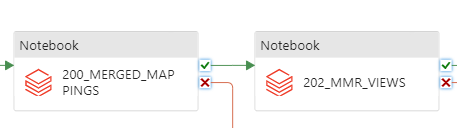
/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/105\_PRODUCT\_MAP\_EXC\_JPN

This notebook is executed for ingestion of Prodcut mapping. As this a common notebook for EU and LA we need to specify geo\_id, in input which are only LA specific. This part is more explained in SOP document under Ad-Hoc ingestion.

A screenshot of a computer

Description automatically generated

1. Execute notebooks:



/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/200\_MERGED\_MAPPINGS

/Shared/CNGC\_COMMON/20\_PREPROCESSING\_VIEWS/202\_MMR\_VIEWS

Which will create necessary tables and views for ingested data.

## 9.3.2.2 MDM\_CM\_Adhoc\_Ingestion:

This pipeline is used to ingest only Customer Mapping data for LA:

The only difference in above explained pipeline is Product mapping notebook will not get executed. Reset all the activities are similar.

## 9.3.2.3 MDM\_PM\_Adhoc\_Ingestion:

This pipeline is used to ingest only Product Mapping data for LA:

The only difference in the above explained pipeline is Customer mapping notebook will not get executed. Reset all the activities are similar.

## Common Layer Pipelines:

## 9.3.3.1 common\_3wd-pipeline

This pipeline ingests the Shipments data on 3wd.

Inner activities:

1. Copy the files from landing zone to incoming

A screenshot of a computer

Description automatically generated

Lookup activity will locate the files which need to be copied along with destination.

A screenshot of a computer

Description automatically generated

Next copy activity will copy the file.

A screenshot of a computer

Description automatically generated

1. Similar activity which will copy the files from incoming to processing folder.
2. Call the common layer orchestration. Which will ingest the shipments data accordingly.

A screenshot of a computer

Description automatically generated

## LA specific pipelines:

1. cngc\_la\_main\_processing\_pipeline
2. cngc\_la\_workday\_orchestration
3. cnos\_gc\_la\_ingestion\_archiving
4. cngc\_la\_smo\_parallel\_execution
5. cngc\_la\_notebook\_execution
6. cngc\_la\_output processing\_without\_DQ\_check
7. cngc\_la\_AAS\_refresh\_without\_DQ\_Check
8. cngc\_la\_output\_refresh\_with DQ
9. LA\_Output\_Data\_Validation\_Split
10. Pipeline\_LA\_MDM\_Ingestion
11. cngc\_la\_NSRD\_GS\_extract\_generation
12. pipeline\_export\_data\_to\_CDL

### 9.3.4.1 cngc\_la\_main\_processing\_pipeline

A screenshot of a computer

Description automatically generated

**Description:** cngc\_la\_main\_processing\_pipeline pipeline used to call all the other activities based on the workday.

**Activity Name**: Start\_ETL\_logging

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_ins\_ETL\_Execution]’ for audit logging.

**Activity Name**: Lookup\_stored\_procedure\_for\_wdp

A screenshot of a computer

Description automatically generated

**Purpose**:

* Uses stored procedure ‘[cngc\_meta].[get\_pwd\_number]’ which fetches positive workday based on current sysdate.

**Activity Name**: Lookup\_stored\_procedure\_for\_wdn

A screenshot of a computer

Description automatically generated

**Purpose:**

* Uses stored procedure ‘[cngc\_meta].[get\_nwd\_number]’ which fetches negative workday based on current sysdate.

**Activity Name**: Set Workday

A screenshot of a computer

Description automatically generated

**Purpose:**

* This activity checks whether the ‘workday’ parameter is empty. If empty it retrieves data from lookup activities: Lookup\_stored\_procedure\_for\_wdp, Lookup\_stored\_procedure\_for\_wdn. If ‘workday’ parameter is not empty its value is returned.

**Activity Name:** split string to array

A screenshot of a computer program

Description automatically generated

**Purpose:**

* This activity uses a split function to separate a string value stored in the ‘workday’ variable into an array based on specified delimiter (in this case, a comma).

**Activity Name**: ForEach\_to\_check\_WD

**A screenshot of a computer

Description automatically generated**

**Purpose**:

* This activity uses switch case within a forEach loop, each case within switch case corresponds to a particular workday, when value is matched with a case it invokes ‘cngc\_la\_workday\_orchestration’ pipeline.

**Activity Name**: ETL\_SucessLog

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_upd\_ETL\_Execution\_Complete]’ for audit logging and provides a record of ETL process executions.

### 9.3.4.2 cngc\_la\_workday\_orchestration

A screenshot of a computer

Description automatically generated with medium confidence

**Description**: In workday pipeline Initially, data will be ingested through various sources based on the given workday parameter and it will process all the general notebooks before processing SMO’s parallelly by running all notebooks in sequence.

**Activity Name**: Start\_ETL\_logging

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_ins\_ETL\_Execution]’ for audit logging.

**Activity Name**: cnos\_gc\_la\_ingestion\_archiving

A screenshot of a computer

Description automatically generated

**Purpose**:

* This Execute Pipeline activity takes WorkDay as input parameter and invokes cnos\_gc\_la\_ingestion\_archiving pipeline for ingesting data based on the corresponding WorkDay parameter.

**Activity Name**: cngc\_la\_general\_notebook\_execution

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Purpose**:

* This Execute pipeline activity invokes cngc\_la\_general\_notebook\_execution pipeline to automate the execution of general notebooks by setting a default parameter as ‘GENERAL’

**Activity Name:** cngc\_la\_smo\_parallel\_execution

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Purpose:**

* This Execute pipeline activity invokes cngc\_la\_smo\_parallel\_execution pipeline to automate the execution of all SMO specific notebooks parallelly.

**Activity Name:** Switch

**A screenshot of a computer

Description automatically generated**

**Purpose:**

* The Switch case activity takes WorkDay parameter as input and automates all the notebooks by giving the default parameter value of SMO as ‘DIMS\_FACTS’ and ‘DQ CHECK’ corresponding to the WorkDay.

**Activity Name**: ETL\_SucessLog

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_upd\_ETL\_Execution\_Complete]’ for audit logging and provides a record of ETL process executions.

### 9.3.4.3 cnos\_gc\_la\_ingestion\_archiving

A screenshot of a computer

Description automatically generated

**Description:**  In the Ingestion phase data are read from various sources based on workday and moved at first to the incoming folder and later to the processing folder.

**Activity Name**: Start\_ETL\_logging

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_ins\_ETL\_Execution]’ for audit logging.

**Activity Name:** Switch\_Ingestion

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | A screenshot of a computer  Description automatically generated |

**Purpose:** Used to ingest/archive data based on workday

* -2WD: -2 workday is used to archive data
* -4WD: On -4WD customer mapping, product mapping, geo mapping and profit center mapping data is ingested
* 3WD: On 3WD simplement data is ingested, we are ingesting glpca, t001, vbrk,vbrp
* 4WD: On 4WD customer mapping, product mapping, geo mapping and profit center mapping data is ingested, NSRD GS report is generated.
* 7WD: On 7WD sharepoint files are ingested, we are ingesting Optima, NIT, and security configuration files.
* 13WD: On 13WD output is processed and AAS Regional model refresh activity is performed.

**Activity Name**: ETL\_SucessLog

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_upd\_ETL\_Execution\_Complete]’ for audit logging and provides a record of ETL process executions.

### 9.3.4.4 cngc\_la\_smo\_parallel\_execution

A screenshot of a computer

Description automatically generated

**Description:** Depending on the workday, this pipeline will process all the SMOs parallelly by running the notebooks in sequence, can be used in scenarios where:

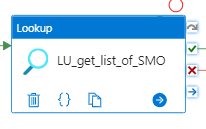
* To process the single/Multiple SMO’s notebooks based on the Execution Day and Subject Area.
* To Process all the SMO’s specific notebooks for a given working day.

**Activity Name**: Start\_ETL\_logging

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_ins\_ETL\_Execution]’ for audit logging.

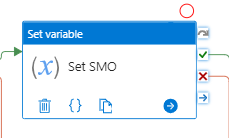
**Activity Name**: LU\_get\_list\_of\_SMO



**Purpose**:

* Used to get distinct SMO from [cngc\_operations].[CNOS\_GC\_ETL\_Notebooks\_LA\_v]

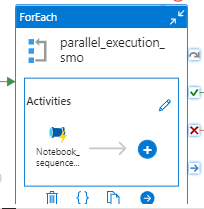
**Activity Name:** Set SMO



**Purpose:**

* This activity is used to retrieve the value of the SMO parameter from the pipeline

**Activity Name:** parallel\_execution\_smo



**Purpose**:

* If variable SMO is empty, it gets list of SMO value from LU\_get\_list\_of\_SMO activity, If SMO does not empty the existing value and executes the ‘cngc\_la\_notebook\_execution’ pipeline for each SMO parallely.

**Activity Name**: ETL\_SucessLog

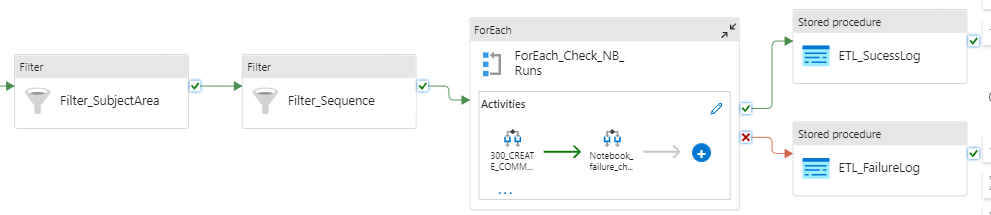
**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_upd\_ETL\_Execution\_Complete]’ for audit logging and provides a record of ETL process executions.

### 9.3.4.5 cngc\_la\_notebook\_execution

A screenshot of a computer

Description automatically generated



**Description**:

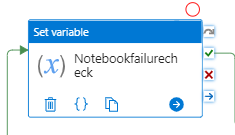
* Notebook sequence execution pipeline is used to execute a sequence of notebooks in a specified order based on the input parameters given.
* It can be used to restart execution from the point of failure and for Ad hoc execution for a specific SMO, execution day and subject area.

**Activity Name**: Start\_ETL\_logging

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_ins\_ETL\_Execution]’ for audit logging.

**Activity name:**  Notebookfailurecheck



**Purpose**:

* Variable notebookfailurecheck is assigned 0 here.

**Activity Name**: Fetch\_All\_Notebooks

A screenshot of a computer

Description automatically generated

**Purpose**:

* Used to select Execution\_Day, SMO, Subject\_Area, Sequence, Notebook\_Name, Notebook\_Path, Execution\_Enabled,ServiceNow\_Location,Last\_Inserted\_Date from metadata table [cngc\_operations].[CNOS\_GC\_ETL\_Notebooks\_LA\_v]
* [cngc\_operations].[CNOS\_GC\_ETL\_Notebooks\_LA\_v] is the main control table where the list of notebooks which are active (Execution Enabled as “Y”) and to be executed are present.

A screenshot of a computer

Description automatically generated

**Activity Name:** Filter\_Execution\_Day

**Purpose:** This activity enables dynamic selection of the execution day based on a provided parameter. It allows conditional filtering of data by comparing the parameter value with the default execution day.

**Activity Name:**  Filter\_SubjectArea

**Purpose:** This activity enables dynamic selection of the SubjectArea based on a provided parameter. Allows filtering of data by comparing the parameter value with the default SubjectArea

**Activity Name:** Filter\_SMOS

**Purpose:** This activity enables dynamic selection of the SMO based on a provided parameter. Allows filtering of data by comparing the parameter value with the default SMO.

**Activity Name:**  Filter\_Sequence

**Purpose:** This activity enables dynamic selection of the Sequence based on a provided parameter. It allows conditional filtering of data by comparing the parameter value with the default Sequence.

**Activity Name:** Notebook\_failure\_check\_condition

A screenshot of a computer

Description automatically generated

**Purpose**:

* This activity is used to determine if a notebook run has failed, by evaluating whether the variable "notebookfailurecheck" is equal to '0'.

A screenshot of a computer

Description automatically generated

**Activity Name**: Notebook\_Execution\_Start\_LA

**Purpose**: Uses stored procedure ‘[cngc\_operations].[sp\_log\_CNOS\_GC\_ETL\_Step\_Hist]’ for writing a log entry before starting notebook execution.

**Activity Name**: Notebook\_run

**Purpose**: This activity is referencing notebook path for running multiple notebooks sequentially.

**Activity Name**: SP\_Log\_Notebook\_Execution\_Success\_LA

**Purpose**: Uses stored procedure ‘[cngc\_operations].[sp\_log\_CNOS\_GC\_ETL\_Step\_Hist]’ for logging successful execution of notebooks.

**Activity Name:** Set NotebookFailureCheck

**Purpose:** Variable notebookfailurecheck is assigned 1 here.

**Activity Name:** SP\_Log\_Notebook\_Execution\_Failure\_LA

**Purpose:**  Uses stored procedure ‘[cngc\_operations].[sp\_log\_CNOS\_GC\_ETL\_Step\_Hist]’ for logging notebooks that failed to execute.

### 9.3.4.6 cngc\_la\_output\_processing\_without\_DQ\_check

**A picture containing screenshot, text, font

Description automatically generated**

**Description:**

* ‘cngc\_la\_output processing\_without\_DQ\_check’ pipeline is used to process the output without executing the DQ checks notebooks.

**Activity Name**: Start\_ETL\_logging

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_ins\_ETL\_Execution]’ for audit logging.

**Activity Name:** cngc\_la\_general\_notebook\_execution

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

**Purpose:**

* This Execute pipeline activity invokes cngc\_la\_general\_notebook\_execution pipeline to automate the execution of general notebooks by setting a default parameter as ‘GENERAL’.

**Activity Name:** cngc\_la\_smo\_parallel\_execution

A screenshot of a computer

Description automatically generated with medium confidence

**A screenshot of a computer

Description automatically generated with medium confidence**

**Purpose:**

* This Execute pipeline activity invokes cngc\_la\_smo\_parallel\_execution pipeline to automate the execution of all SMO specific notebooks parallelly.

**Activity Name:** cngc\_la\_notebook\_execution\_DIM\_FACT\_4WD

A picture containing text, screenshot, font, line

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Purpose:

* This Execute pipeline activity invokes cngc\_la\_general\_notebook\_execution pipeline to automate the execution of dims\_facts notebooks by setting a SMO default parameter as ‘DIMS\_FACTS’ and EXECUTION\_DAY default parameter as ‘13WD’.

**Activity Name:** cngc\_la\_notebook\_execution\_DQ\_CHECKS\_4WD

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence

**Purpose:**

* This Execute pipeline activity invokes cngc\_la\_general\_notebook\_execution pipeline to automate the execution of DQ checks notebooks by setting a SMO default parameter as ‘DQ\_CHECKS’ and EXECUTION\_DAY default parameter as ‘13WD’.

**Activity Name:** cngc\_la\_output\_refresh\_without\_DQ\_Check\_copy1

A picture containing text, font, line, white

Description automatically generated

**Purpose:**

* This Execute pipeline activity invokes ‘cngc\_la\_AAS\_refresh\_without\_DQ\_Check’ pipeline to refresh the outputs before executing DQ checks notebooks.

**Activity Name**: ETL\_SucessLog

**Purpose**:

* Uses stored procedure ‘[cngc\_operations].[usp\_upd\_ETL\_Execution\_Complete]’ for audit logging and provides a record of ETL process executions.

### 9.3.4.7 cngc\_la\_AAS\_refresh\_without\_DQ\_check

**A screenshot of a computer

Description automatically generated with low confidence**

**A screenshot of a computer

Description automatically generated with low confidence**

**Description:**

* The ‘cngc\_la\_AAS\_refresh\_without\_DQ\_Check’ pipeline executes all pipelines required for LA output processing before processing the DQ checks.

**Activity Name:** pipeline\_get\_outgng\_la\_pea\_fct\_table\_list.

A picture containing text, font, line, white

Description automatically generated

**Purpose:**

* The Execute pipeline activity invokes ‘pipeline\_get\_outgng\_la\_pea\_fct\_table\_list’ pipeline to get DIM, FCT table list - filtered for LA.

**Activity Name:** pipeline\_export\_la\_pea\_output\_sqlserver2blob

A picture containing text, font, line, screenshot

Description automatically generated

**Purpose**:

* The Execute pipeline activity invokes ‘pipeline\_export\_la\_pea\_output\_sqlserver2blob’ pipeline to get DIM, FCT table list and moves it from SQL server to blob.

**Activity Name**: call\_pipeline\_get\_outputs2phcdashboard.

A screenshot of a computer

Description automatically generated

**Purpose**:

* Lookup activity will fetch the metadata I.e viewname and blob path
* Copy activity will copy the views from fetched blob path.

**Activity Name**: call\_pipeline\_get\_outputs2phcdashboard.

A screenshot of a computer

Description automatically generated

**Purpose**:

* Copy activity will copy the output in cngc-outgoing blob.

**Activity Name**: call\_pipeline\_output\_to\_sqlserver

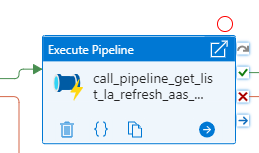
A screenshot of a computer

Description automatically generated

**Purpose**:

* Scale up SQL server
* Lookup activity will Fetch the metadata consist of table names and blob path for copying
* Copy activity will copy the data from blob to SQL tables.

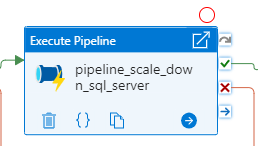
**Activity Name**: call\_pipeline\_get\_list\_la\_refresh\_aas\_model.



**Purpose**:

* Will refresh AAS server depending upon the data factory environment.

**Activity Name**: pipeline\_scale\_down\_sql\_server



**Purpose**:

* Will scale down the SQL server.

**Activity Name**: pipeline\_scale\_down\_sql\_server

A screenshot of a computer

Description automatically generated

**Purpose**:

* Will refresh the PHC output and execute the CDL scripts.

**Activity Name**: Wait-LA-15WD.

A screenshot of a computer

Description automatically generated

**Purpose**:

* Wait activity

### 9.3.4.8 LA\_Output\_Data\_Validation\_Split

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence

**Activity Name**: pipeline\_scale\_up\_sql\_server

A screenshot of a computer

Description automatically generated with low confidence

**Purpose**:

* The Execute pipeline activity invokes ‘pipeline\_scale\_up\_sql\_server’ pipeline to scale up SQL server.

**Activity Name**: LA\_DQ\_Checks\_Common\_Notebook

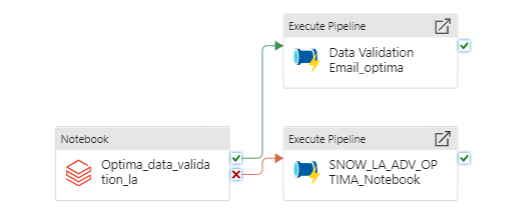
A screenshot of a computer

Description automatically generated with medium confidence

**Purpose**:

* The notebook executes a specific notebook located at the path ‘CNGC\_LA\_SMO\_SPLIT\_PROJECT/10\_AUTOMATED\_DQ\_CHECKS/ADV\_00\_LA DQ\_CHECK\_Common\_Code’.
* This notebook contains all the common code required to run the LA DQ checks.

**Activity Name**: execute\_optima\_dq\_checks.



**Purpose:**

* The Execute pipeline activity invokes ‘Optima\_Data\_validation’ pipeline to execute the optima DQ checks notebooks at path ‘/Shared/CNGC\_LA/60\_DQ\_CHECKS/ADV\_03\_DQ\_CHECK\_OPTIMA\_FUND\_ID\_MAPPING’.
* Sends latest testcase execution results of OPTIMA.

**Activity Name:** LA\_Input\_vs\_Output\_DQ\_Check

A screenshot of a computer

Description automatically generated with medium confidence

**Purpose:**

* The notebook presents in this path ‘/shared/CNGC\_LA/10\_AUTOMATED\_DQ\_CHECKS/ADV\_01\_LA\_DQ\_CHECK\_INPUT\_VS\_OUTPUT’ is executed which contains all Input vs Output DQ checks for LA KPIs. The notebook contains code for:

1. NIT Ingested values
2. SD Fixed budget Ingested values
3. SD live rates Ingested values
4. SD TPR Ingested values
5. FMR Ingested values
6. CMI Ingested values

**Activity Name:** LA\_Basic\_Output\_DQ\_Checks

A screenshot of a computer

Description automatically generated with low confidence

**Purpose:**

* The notebook presents in this path ‘/CNGC\_LA\_NEW/10\_AUTOMATED\_DQ\_CHECKS/ADV\_02\_LA\_DQ\_CHECK\_BASIC\_OUTPUT\_CHECKS’ is executed. This notebook contains code for

1. Output Duplicate check
2. Output MD Completeness check
3. NOS Completeness check

**Activity Name**: LA\_NIT\_Valid\_RDS\_Customers

A screenshot of a computer

Description automatically generated with low confidence

**Purpose:**

* The notebook presents in this path ‘/CNGC\_LA\_NEW/10\_AUTOMATED\_DQ\_CHECKS/ADV\_04\_LA\_DQ\_CHECKS\_RAW\_INPUT’ is executed.

**Activity Name**: Data Validation Email

A picture containing text, font, screenshot, line

Description automatically generated

**Purpose:**

* The Execute pipeline activity invokes ‘Data\_Validation\_LA’ pipeline which sends latest testcase execution status.

**Activity Name:** ServiceNOW\_for\_Test\_Case\_Failures

A picture containing text, font, screenshot, line

Description automatically generated

**Purpose:**

* The Execute pipeline activity invokes ‘LA\_ADV\_Service\_Now\_Integration’ pipeline to generate an incident in case of data validation failure.

**Activity Name:** ETL Success Log

**Purpose:**

* Uses stored procedure ‘[cngc\_operations].[usp\_upd\_ETL\_Execution\_Complete]’ for audit logging and provides a record of ETL process executions.

### 9.3.4.9 cngc\_la\_output\_refresh\_with\_DQ



**Activity Name**: Call\_LA\_Output\_Data\_Validation

A screenshot of a computer

Description automatically generated

**Purpose:**

* Used to call LA\_Output\_Data\_Validation\_Split Pipeline.

**Activity Name**: Check\_LA\_DQ\_Checks\_Outcome

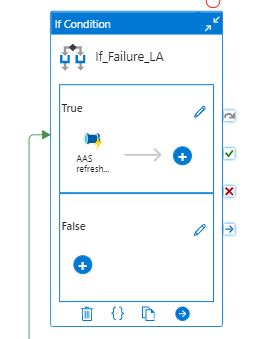
A screenshot of a message

Description automatically generated

**Purpose**:

* The lookup activity will fetch the count of error in previously run DQ checks depending on the latest run

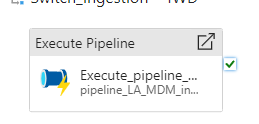
**Activity Name**: If\_Failure\_LA



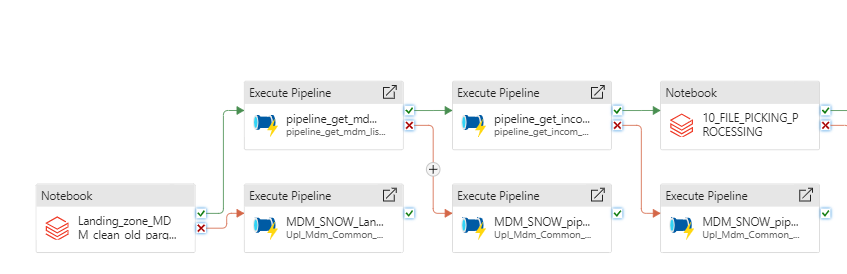
**Purpose:**

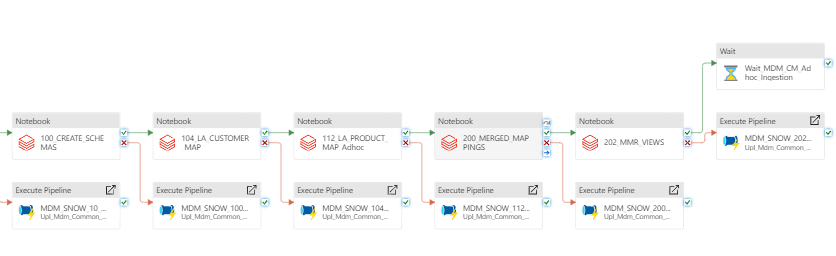
* If an activity consists of previously run lookup activity count, upon satisfying the if condition AAS will get refreshed.

### 9.3.4.10 Pipeline\_LA\_MDM\_Ingestion



Inner Components:





Purpose:

The general purpose of the above stated pipeline is to ingest the MDM data. The data MDM sources in this scenario are Customer and Product mapping. At high level these pipelines first copy the data from landing zone in cngc-storage and put the data in processing folder.

The DB notebooks associated with this pipeline create the schemas and tables by reading the parquet files present in processing folder.

**Notebooks used are:**

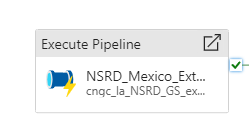
/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/10\_FILE\_PICKING\_PROCESSING

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/100\_CREATE\_SCHEMAS

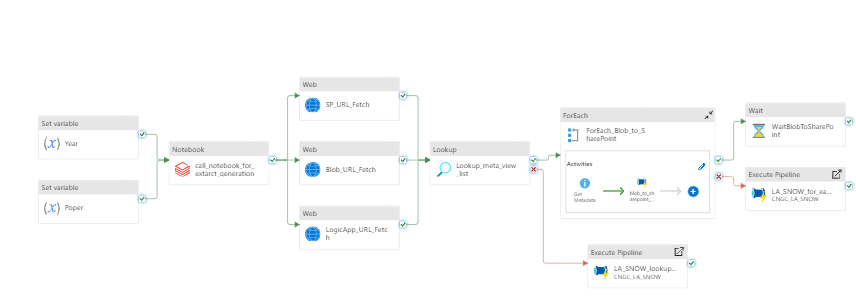
/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/104\_LA\_CUSTOMER\_MAP

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/105\_PRODUCT\_MAP\_EXC\_JPN

### 9.3.4.11 cngc\_la\_NSRD\_GS\_extract\_generation



Inner activities:



Purpose:

This pipeline runs the notebook:

/Shared/CNGC\_LA\_SMO/13\_MANUAL\_EXTRACTS/SAP Extracts for Gross Sales

To generate SAP extracts for gross sales, which are generated at blob storage, and further copied to SharePoint location.

### 9.3.4.12 pipeline\_export\_data\_to\_CDL

A screenshot of a computer

Description automatically generated

Purpose:

The purpose of this pipeline is to export the data at PHC level.

# ADB COMPONENTS (Tables and views created)

## 10\_INGESTION

The 10\_INGESTION has the following notebooks for data extraction purposes.

### 10.3.1 DATA Extraction from share point(106\_SHAREPOINT\_FILES\_<SMO\_NAME>):

* Data is extracted from files for Customer Mapping from uploader, and Security Configuration files.
* Security configuration files are stored at blob location ‘dbfs:/mnt/cngc-storage/processing/LA\_security\_config\_file/Security Configuration.xlsx’.
* Manual customer mapping is stored at blob location ‘dbfs:/mnt/cngc-storage/processing/manual\_customer\_mapping/LA\_AZ\_Manual\_Customer\_Mapping.xlsx’.

### 10.1.2 Processing and creating Simplement data from Storage Account(107\_SIMPLEMENT\_DATA\_<SMO\_NAME>):

* Simplement Data extraction is utilized for NSRD calculations, Currency mapping, Customer mapping.
* The table ‘stage\_la\_simplement.glpca\_cdl\_<SMO\_NAME>\_BASE’ is created by reading data from files located at **'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/GLPCA'** and a new table is created called stage\_la\_simplement.glpca\_cdl\_ **<SMO\_NAME>** based on the previous table, but only selecting data where **‘rbukrs’** column equals **‘<SMO\_SPECIFIC\_VALUE>’**.
* The table **vbrp\_cdl\_SOURCE\_<SMO\_NAME>** is created by reading data from files located at **'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/vbrp/'** and a new table is created called stage\_la\_simplement.vbrp\_cdl\_<SMO\_BASE>\_BASE based on the previous table, but only selecting data where **‘lland\_auft’** column equals **‘SMO\_SPECIFIC\_VALUE’**.
* The table **‘t001\_cdl\_SOURCE\_<SMO\_NAME>’** is created by reading data from files located at **'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/ t001/'** and a new table is created called **‘t001\_cdl\_SOURCE\_<SMO\_NAME>’** based on the previous table, but only selecting data where **‘bukrs\_glob’** column equals **‘<SMO\_SPECIFIC\_VALUE>’**.
* The table **‘**stage\_la\_simplement.t001\_cdl\_ARGENTINA\_BASE\_**<SMO\_NAME>’** is created by reading data from files located at **'dbfs:/mnt/cngc-storage/processing/simplement\_cdl/glpca\_backup'** and a new table is created called **‘glpca\_backup\_<SMO\_NAME>’** based on the previous table, but only selecting data where **‘rbukrs’** column equals **‘<SMO\_SPECIFIC\_VALUE>’**.
* These exact same tables have been created for the rest of the SMOs as well, but with the corresponding SMO name and SMO specific filters applied.

### 10.1.3 Extracting Cust & Prod Key from storage account(108\_LA\_DATA\_HUB\_<SMO\_NAME>):

* Two tables are created **‘cust\_keys\_<SMO\_NAME>’** reading data from files located at ‘dbfs:/mnt/cngc-storage/processing/la\_blob\_surrogate\_keys/cust’ and **‘prod\_keys\_<SMO\_NAME>’** reading data from files located at ‘dbfs:/mnt/cngc-storage/processing/la\_blob\_surrogate\_keys/prod’ .
* ‘Customer\_mapping\_<SMO\_NAME>’ table is created from the raw ‘la\_customer\_mapping\_upd’ table.
* These are created to map customer and product keys for the SMO’s.
* The following filters are applied to extract only LA region-specific data:
* GEO = <SMO\_NAME>
* These exact same tables have been created for the rest of the SMOs as well, but with the corresponding SMO name and SMO specific filters applied.

### 10.1.4 Exporting Customer mapping data to CSV (110\_CREATE\_CSV\_TO\_SP\_<SMO\_NAME>):

* View **‘nsrd\_gross\_sales\_customer\_mapping\_only\_v\_<SMO\_NAME>’** is created which selects distinct data from union of two tables and filters records by fiscal year based on subquery from **‘stage\_common. SODE\_CAL\_FDIM’.**
* **‘customer\_mapping\_analysis\_<SMO\_NAME>’** table is the combination of data coming from 3 different sources MDM, SHIP and SAP.
* Customer Mapping from Data HUB is joined with SAP & Shipments data to categorize the Cust L9 IDs based on measures the Customer is associated with, the possible scenarios are:

1. HUB-data comes from only MDM, no corresponding match found in shipments and sap.
2. HUB\_SHIP -data is there in MDM and shipment but not present in sap, we will not receive Gross Sales and NSRD for these Customers.
3. SAP - customerid exists in sap but is not found in MDM and shipments.
4. SAP\_HUB-customer has sap data, also present in MDM but no shipments.
5. SAP\_HUB\_SHIP-These are the full match of customer in all the systems and can be considered for the customer mapping base. It has sap data such as gross sales and NSRD also customer is there in MDM and shipments.

* **‘stage\_la\_excels.manual\_customer\_mapping\_automated\_<SMO\_NAME>’** is created using the **‘customer\_mapping\_analysis\_<SMO\_NAME>’** table by filtering records with value of ‘DESCRIPTION’ is either ‘MDM\_SHIP' or 'SAP\_MDM\_SHIP'.
* stage\_common.manual\_customer\_mapping\_dim\_<smo\_name> table is created by joining goeo\_mapping to obtain fommted CUSTOMER\_MAPPING\_KEY
* **Stage\_la\_excels.manual\_customer\_mapping\_automated\_export\_<SMO\_NAME>** is extarcted to blob storage path: dbfs:/mnt/cngc-storage/dim\_fct\_output/uploader\_export\_manual\_customer\_mapping\_v\_<SMO\_NAME>
* Lastly python code is written to copy the upadated cutomer mapping file to blob storage account.

### 10.1.5 Creating Geo Mapping(111\_CREATE\_FIX\_MAPS\_<SMO\_NAME>):

* Table **geo\_56\_mapping\_<SMO\_NAME>** is created and refreshed by specifying a hard coded value such as:
* CAST (954 AS INT)               GEO\_MAP\_ID,
* CAST (5 AS INT)               GEO\_MAP\_LEVEL,
* CAST ('09429' AS CHAR (5))       GEO\_MAP\_PROFIT\_CNTR
* These exact same tables have been created for the rest of the SMOs as well, but with the corresponding SMO name and SMO specific filters applied.

## 20\_PREPROCESSING\_VIEWS

The 20\_PREPROCESSING VIEWS has following notebooks inside it:

### 10.2.1 Creating Prod and Cust Keys(204\_LA\_DATAHUB\_VIEWS\_<SMO\_NAME>):

* **‘cust\_keys\_v\_<SMO\_NAME>’** table is created selecting only ‘cust key’ and ‘ship-to’ columns from **‘cust\_keys\_<SMO\_NAME>’** table which is already filtered to SMO specific data.
* Similarly, **‘prod\_keys\_v\_<SMO\_NAME>’** table is created selecting only ‘prod key’ and ‘fpc’ columns from **‘prod\_keys\_<SMO\_NAME>’** table which is already filtered to SMO specific data.
* These exact same tables have been created for the rest of the SMOs as well, but with the corresponding SMO name and SMO specific filters applied.

### 10.2.2 Extraction of Optima data and Uploader data(205\_RLS\_OPTIMA\_NIT\_CM\_VIEWS\_<SMO\_NAME>):

* Data is extracted from Optima for SD and data for NIT is extracted from Uploader and added necessary filters to extract only SMO specific data.
* stage\_la\_excels.manual\_customer\_mapping\_v\_<SMO\_NAME> view is created from stage\_la\_excels.manual\_customer\_mapping\_automated\_<SMO\_NAME>
* ‘manual\_nit\_v\_<SMO\_NAME>’ table is created to extract NIT data directly from uploader ‘stage\_uploader.la\_nit’ with filter ‘original\_le=<SMO\_SPECIFIC\_VALUE>’ for eg. Brazil orig\_le='682' to extract only SMO specific data.
* A table ‘**sd\_fixed\_budget\_la\_v\_<SMO\_NAME>’** is createdby selecting columns from Optima through file name - **DATA\_EXTRACT\_FUND\_ADJUSTOMENT\_TRANSFER**
* A table ‘sd\_live\_rates\_la\_v\_<SMO\_NAME>’ is created by selecting columns from table stage\_uploader.sd\_live\_rates\_la\_<SMO\_NAME>
* Filters used product\_level <> 'Unknown'

### 10.2.3 Extraction of Simplement data(206\_SIMPLEMENT\_VIEWS\_<SMO\_NAME>):

* Data is extracted from simplement glpca for NSRD and gross sales kpi.
* **‘glpca\_v\_<SMO\_NAME>’** table is created by doing union of **‘glpca\_cdl\_<SMO\_NAME>’** and **‘glpca\_backup\_la\_v\_<SMO\_NAME>’** and following filters are applied:
* RBUKRS which is smo specific
* simp\_chng\_type\_code <> 'D'
* RYEAR
* RRCTY = '0' AND RVERS = '000' AND KOKRS = 'XX00'
* RACCT BETWEEN '0030000000' AND '0030009999'
* (RYEAR,POPER)IN ('2021001','2021002','2021003','2021004','2021005','2021006','2021007','2021008','2021009','2021010','2021011','2021012','2022001','2022002','2022003','2022004','2022005','2022006','2022007','2022008')
* **‘vbrp\_v\_<SMO\_NAME>’** table is created selecting only ‘KUNWE’ and ‘VBELN’ columns from **‘vbrp\_cdl\_<SMO\_NAME>’**.
* **‘T001\_v\_<SMO\_NAME>’** table is created from **‘t001\_cdl\_<SMO\_NAME>’** with filters: WAERS is not null and WAERS <> ‘‘and WAERS <> ' '
* **‘gross\_saless\_account\_list\_v\_<SMO\_NAME>’** is created for selecting all the account id’s related to gross sales from **‘mnr\_acct\_id\_type\_mapng\_dim\_vw\_v’**.
* Similarly, **‘account\_nsrd\_type\_map\_v\_<SMO\_NAME>’** selects all the account id’s having NSRD type from **‘mnr\_acct\_id\_type\_mapng\_dim\_vw\_v’** with filter NVL(NSRD\_TYPE\_NAME,'‘) <>''
* stage\_la\_simplement.nsrd\_gross\_sales\_la\_customer\_mapping\_only\_v\_<SMO\_NAME> view is created from view stage\_la\_simplement.glpca\_v\_<smo\_name> by joining with legel entity mapping view stage\_mmr.mnr\_le\_mapng\_dim\_vw\_v and geo mapping stage\_mmr.mnr\_geo\_mapng\_dim\_vw\_v and stage\_mmr.gross\_saless\_account\_list\_v profit center mapping stage\_common.SODE\_PC084\_FDIM

## 40\_PROCESSING\_REPORTING

The 40\_PROCESSING\_REPORTING has following notebooks in it:

### 10.3.1 Creation of manual customer input(400\_CREATE\_LADMAR\_INPUTS\_TC\_TO\_LC\_<SMO\_NAME>):

* stage\_work\_la.manual\_customer\_input\_unpivoted\_pc\_<SMO\_NAME> IS CREATED from customer mannual input stage\_mmr.mnr\_cust\_manul\_input\_dim\_vw\_v by adding profit center valure from sub query.also by doing union of two selects which are taken from two different profit centre 09427 and 09429. The filter applied for profit centre 09427 select is nvl(NSRD\_MANUL\_INPUT\_LC\_BASE,0)<>0 or nvl(SD\_MANUL\_INPUT\_LC\_BASE,0)<>0 while for profit center 09429 select, the filter applied is nvl(NSRD\_MANUL\_INPUT\_LC\_EXPORT,0)<>0 or nvl(SD\_MANUL\_INPUT\_LC\_EXPORT,0)<>0
* A view stage\_work\_la.manul\_customer\_input\_v <SMO\_NAME> is created by selecting necessary columns from manual\_customer\_input\_unpivoted\_pc\_<SMO\_NAME>.

### 10.3.2. Processing of direct shipments(500\_LA\_DIR\_SHIP\_FACT\_PROCESS\_<SMO\_NAME>):

* View ‘ship\_step\_100\_sode\_hist\_star\_work\_v\_<SMO\_NAME>’ selects data directly from the ‘ship\_step\_100\_sode\_hist\_star\_work\_tc\_lc’, joins with ‘stage\_la\_fix\_map.geo\_56\_mapping\_<SMO\_NAME>’ with the following applied filters: (NVL(GROSS\_TC\_AMT,0)<>0 OR NVL(STAT\_UNIT\_QTY,0)<>0 ) and ORG\_ID != '9999' and prod\_csu\_type\_code<>'D' and CORP\_OFFCL\_SHIP\_FLAG='Y'.
* stage\_la\_simplement.sd\_tpr\_aggr\_fct\_input\_<smo-name>is created from source repo\_la.sd\_tpr\_<sm-name> by cross joining with stage\_mmr.mnr\_geo\_mapng\_dim\_vw\_unified\_v used to select profit center.
* The table ’all\_aggr\_fct\_input\_<SMO\_NAME>’ is a combination of data from 3 other tables:’nit\_aggr\_fct\_input\_<SMO\_NAME>’,’nsrd\_gross\_sales\_aggr\_fct\_input\_<SMO\_NAME>’,’ sd\_tpr\_aggr\_fct\_input\_ARGENTINA’ with added filters ‘GEO\_ID’ in each table.
* stage\_work\_la.ship\_step\_110\_artifical\_star\_work\_<smo\_name> is created from source stage\_la\_simplement.all\_aggr\_fct\_input\_<SMO\_NAME>

INNER JOIN stage\_common.SODE\_CAL\_FDIM

INNER JOIN stage\_common.SODE\_PC064\_FDIM PC064

INNER JOIN stage\_common.SODE\_C898\_FDIM

INNER JOIN stage\_common.SODE\_P5005\_FDIM

LEFT JOIN stage\_common.SODE\_P5801\_FDIM

INNER JOIN stage\_common.SODE\_G705\_FDIM

LEFT JOIN stage\_la\_fix\_map.geo\_56\_mapping\_<SMO\_NAME>

* Next table is created stage\_work\_la.ship\_step\_120\_artifical\_star\_work\_<SMO\_NAME> is created by using source stage\_work\_la.ship\_step\_110\_artifical\_star\_work\_<SMO\_NAME> and joining with LEFT JOIN stage\_la\_fix\_map.geo\_56\_mapping\_<SMO\_NAME> LEFT OUTER JOIN stage\_work\_la.ship\_step\_100\_sode\_hist\_star\_work\_v\_<SMO\_NAME>
* TABLE stage\_work\_la.ship\_step\_130\_artifical\_base\_union\_work\_<SMO\_NAME> is created from source table stage\_work\_la.ship\_step\_120\_artifical\_star\_work\_<SMO\_NAME> by union with stage\_work\_la.ship\_step\_100\_sode\_hist\_star\_work\_v\_<SMO\_NAME>
* Table stage\_work\_la.nsrd\_gs\_prev\_year\_fct\_1ddefault\_<SMO\_NAME> is created by using stage\_la\_simplement.nsrd\_gross\_sales\_v\_<SMO\_NAME> and joining with

LEFT OUTER JOIN stage\_mmr.mnr\_pc\_sbstr\_nsrd\_dim\_vw\_v,

LEFT OUTER JOIN stage\_mmr.mnr\_le\_mapng\_dim\_vw\_v,

LEFT OUTER JOIN stage\_la\_simplement.account\_nsrd\_type\_map\_v\_<SMO\_NAME>

LEFT OUTER JOIN stage\_la\_simplement.gross\_saless\_account\_list\_v\_<SMO\_NAME>

INNER JOIN stage\_common.SODE\_CAL\_FDIM

INNER JOIN stage\_work\_la.ship\_step\_100\_sode\_hist\_star\_work\_v\_<SMO\_NAME>

* Table stage\_work\_la.nsrd\_gs\_prev\_year\_fct\_1ddefault\_no\_curr\_<SMO\_NAME> is created from stage\_work\_la.nsrd\_gs\_prev\_year\_fct\_1ddefault\_<SMO\_NAME> by joing with LEFT OUTER JOIN stage\_work\_la.ship\_step\_100\_sode\_hist\_star\_work\_v\_<SMO\_NAME>

Table stage\_work\_la.nsrd\_gs\_prev\_year\_fct\_subsector\_<SMO\_NAME> is created from view stage\_la\_simplement.nsrd\_gross\_sales\_v\_<SMO\_NAME> by joing with the different views and tables.

LEFT OUTER JOIN stage\_mmr.mnr\_pc\_sbstr\_nsrd\_dim\_vw\_v

LEFT OUTER JOIN stage\_mmr.mnr\_le\_mapng\_dim\_vw\_v

LEFT OUTER JOIN stage\_la\_simplement.account\_nsrd\_type\_map\_v\_<SMO\_NAME>

LEFT OUTER JOIN stage\_la\_simplement.gross\_saless\_account\_list\_v\_<SMO\_NAME>

INNER JOIN stage\_common.SODE\_CAL\_FDIM

INNER JOIN stage\_work\_la.ship\_step\_100\_sode\_hist\_star\_work\_v\_<SMO\_NAME>

* ‘ship\_step\_140\_artifical\_base\_nsrd\_work\_<SMO\_NAME>’ is created by selecting data from three different tables:(‘w’,’w\_nsrd’,’w\_nsrd\_subsect’) ‘ship\_step\_130\_artifical\_base\_union\_work\_<SMO\_NAME>’, ‘nsrd\_gs\_prev\_year\_fct\_1ddefault\_no\_curr\_<SMO\_NAME>’, ‘nsrd\_gs\_prev\_year\_fct\_subsector\_no\_curr\_<SMO\_NAME>’ using the COALESCE function to select first non-null value.
* **‘dir\_ship\_la\_fct\_lc\_presum\_<SMO\_NAME>’** is created using **‘ship\_step\_140\_artifical\_base\_nsrd\_work\_<SMO\_NAME>’** and joining with filter table that filters **‘mnr\_proft\_ctr\_mkt\_mapng\_dim\_la\_mv’** table to only include records where ‘RDS\_GEO\_NAME’ = 'LATIN AMER AREA'.
* The ‘dir\_ship\_la\_fct\_join\_curr\_<SMO\_NAME>’ table selects distinct columns from three different tables: (‘ship\_lc’, ’ship\_cusd’, ’ship\_rusd’); ‘dir\_ship\_la\_fct\_lc\_<SMO\_NAME>’,’ dir\_ship\_la\_fct\_cusd\_v\_<SMO\_NAME>’,’ dir\_ship\_la\_fct\_rusd\_v\_<SMO\_NAME>’.It Left joins the ‘ship\_lc’ with ‘ship\_cusd’ and ‘ship\_rusd’.

### 10.3.3 Processing TDC data(550\_LA\_TDC\_FACT\_PROCESS\_<SMO\_NAME>):

* TDC value is always provided in USD, A table ‘mnr\_tdc\_su\_v\_<SMO\_NAME>’ is created from source ‘mnr\_tdc\_su\_dim\_vw\_v’ inner join ‘mnr\_geo\_mapng\_dim\_vw\_unified\_v’ with filters such as TDC\_USD\_SU <> 0 , REGN\_NAME = 'LA' and GEO\_ID
* A table ‘tdc\_fpc\_lvl’\_<SMO\_NAME> is created from ‘dir\_ship\_la\_fct\_<SMO\_NAME>’ left join ‘repo\_la.mnr\_tdc\_su\_v<SMO\_NAME>’ with condition f.CURRENCY\_ID NOT IN ('CUSD','RUSD')
* Table ‘tdc\_brand\_lvl\_<SMO\_NAME>’ is created from ‘tdc\_fpc\_lvl\_<SMO\_NAME>’ with conditions TDC\_PER\_SU\_FPC IS NOT NULL AND TDC\_PER\_SU\_FPC <> 0 AND GEO\_ID='GEO\_ID'
* Table ‘tdc\_category\_lvl\_<SMO\_NAME>’ is created from ‘tdc\_fpc\_lvl\_<SMO\_NAME>’ with filters TDC\_PER\_SU\_FPC IS NOT NULL AND TDC\_PER\_SU\_FPC <> 0 AND GEO\_ID='GEO\_ID'
* The table ‘tdc\_all\_prod\_levels\_<SMO\_NAME>’ is generated by performing a left join operation between the tables ‘tdc\_fpc\_lvl\_<SMO\_NAME>’ and ‘tdc\_brand\_lvl\_<SMO\_NAME>’ table and ‘tdc\_category\_lvl\_<SMO\_NAME>’.
* View stage\_mmr.mnr\_fx\_rate\_tdc\_dim\_vw\_max\_dt\_v\_<SMO\_NAME> from stage\_mmr.mnr\_fx\_rate\_tdc\_dim\_vw\_v by joining with fx rates master data from CDL from stage\_dir\_shipments.masterdata\_exchg\_rate\_fct .
* View repo\_la.tdc\_amt\_la\_lc\_<SMO\_NAME> from source repo\_la.tdc\_all\_prod\_levels\_<SMO\_NAME> and joining with

INNER JOIN stage\_mmr.mnr\_geo\_mapng\_dim\_vw\_unified\_v

INNER JOIN stage\_mmr.mnr\_fx\_rate\_tdc\_dim\_vw\_max\_dt\_v\_<SMO\_NAME>

* Table repo\_la.tdc\_la\_fct\_<SMO\_NAME> from table repo\_la.tdc\_amt\_lc\_<SMO\_NAME>

repo\_la.tdc\_amt\_rusd\_<SMO\_NAME>

repo\_la.tdc\_amt\_cusd\_<SMO\_NAME>

LEFT OUTER JOIN stage\_la\_excels.rls\_security\_v\_<SMO\_NAME> to obatined the rls security.

* At last table ‘tdc\_la\_fct\_join\_curr\_<SMO\_NAME>’ is created by performing left join between the table ‘repo\_la.tdc\_amt\_lc\_<SMO\_NAME>’ and the tables ‘tdc\_amt\_cusd\_<SMO\_NAME>’ , ‘repo\_la.tdc\_amt\_rusd\_<SMO\_NAME>’ and ‘rls\_security\_v\_<SMO\_NAME>’ to join different currency types and add rls\_key.

### 10.3.4 Processing PEA data(560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>)

* ‘pea\_shipment\_agg\_<SMO\_NAME>’ table is used to create shipment aggregated view using table ‘dir\_ship\_la\_fct\_lc\_<SMO\_NAME>’ with a geoid filter
* A TDC input table ‘pea\_mnr\_tdc\_su\_<SMO\_NAME>’ is created from ‘mnr\_tdc\_su\_dim\_vw\_v’ based on where condition GEO\_ID='GEO\_ID'
* Table repo\_la.pea\_mnr\_tdc\_su\_ARGENTINA from view stage\_mmr.mnr\_tdc\_su\_dim\_vw\_v
* Table ‘pea\_data\_join\_<SMO\_NAME>’ is created from tables ‘prod\_fpc\_gtin\_assoc’ Cross join ‘pea\_shipment\_agg\_<SMO\_NAME>’ to get fiscal years available in shipments. Tables ‘pea\_shipment\_agg\_<SMO\_NAME>’ and ‘pea\_mnr\_tdc\_su\_<SMO\_NAME>’ is combined to get shipments and TDC data.
* Following filters were used:

PROD\_H5005\_CURR\_IND = 'Y'

GEO\_ID=SMO wise which is SMO specific.

* Created ‘pea\_gtin\_avg\_<SMO\_NAME>’ from repo\_la.pea\_data\_join\_<SMO\_NAME>by calculating average TDC per GTIN by joining with stage\_common.prod\_fpc\_gtin\_assoc.
* A table ‘pea\_gtin\_level\_tdc\_avg<SMO\_NAME>’ is created by applying enrich logic for all hierarchy level,
* when TDC\_CU from CU\_GTIN level is positive - apply
* when TDC\_SU from CU\_GTIN level is NULL, equal to zero or negative - apply Brand Form average
* when TDC\_SU average from Brand Form level is NULL, equal to zero or negative - apply Segment average
* when TDC\_SU average from Segment level is NULL, equal to zero or negative - apply Brand average
* when TDC\_SU average from Brand level is NULL, equal to zero or negative - apply Category average
* when TDC\_SU average from Category level is equal to zero or negative - put NULL value as TDC\_SU
* Created ‘pea\_product\_levels\_tdc\_avg\_<SMO\_NAME>’ CATEGORY/BRAND/SEGMENT/BRAND FORM level by applying below enrich logic
* for TDC\_SU for a given product level
* when TDC\_SU for a given product level is NULL, negative or zero, apply average from higher level, repeat until first positive (and non-nullable by this) average is found
* if TDC\_SU average for a category (highest level considered) is negative 0 - apply NULL
* Created ‘pea\_la\_fct\_before\_geo\_filter\_<SMO\_NAME>’ for updating timestamp for each load.
* Table ‘pea\_la\_fct\_<SMO\_NAME>’ creates PEA Fact table from ‘pea\_la\_fct\_before\_geo\_filter\_<SMO\_NAME>’ by applying filters geoid and REGION='LA'.
* More specific information would be found in PEA section.

### 10.3.5 Processing NIT data(600\_LA\_NIT\_FACT\_PROCESS\_<SMO\_NAME>):

* ‘nit\_aggr\_la\_v\_<SMO\_NAME>’ view selects data from ‘manual\_nit\_v\_<SMO\_NAME>’ and ‘manual\_customer\_mapping\_v\_<SMO\_NAME>’ tables and joins them with ‘SODE\_PC084\_FDIM’ and ‘MNR\_LE\_MAPNG\_DIM\_MV’ tables. The view is filtered to only include data with values specific to each SMO: orig\_le, CUSTM\_SMO\_NAME, proft\_ctr\_3\_id and GEO.
* Table repo\_la.nit\_la\_fct\_lc\_la\_only\_<SMO\_NAME> is created from stage\_la\_excels.nit\_aggr\_la\_v\_<SMO\_NAME> by Joining with repo\_la.dir\_ship\_la\_fct\_ARGENTINA and stage\_mmr.mnr\_sap\_shipt\_mapng\_dim\_vw\_v.
* ‘nit\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME>’ view selects data from table ‘nit\_la\_fct\_lc\_<SMO\_NAME>’ and converts currency from original currency to USD using the exchange rates from ‘frgn\_exchg\_rate\_dim\_v’ table.
* ‘nit\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ retrieves data from ‘nit\_la\_fct\_lc\_<SMO\_NAME>’ and joins with table ‘FX\_mth\_multiple\_currencies\_avg\_closed\_fy\_v’ on the condition that trgt\_iso\_crncy\_code=’USD’ and srce\_iso\_crncy\_code is equal to CURRENCY\_ID of ‘nit\_la\_fct\_lc\_<SMO\_NAME>’.The GIV\_AMT and NIV\_AMT are calculated by multiplying GIV\_AMT and NIV\_AMT values with corresponding exchange rates.
* ‘nit\_la\_fct\_<SMO\_NAME>’ created by selecting data from three different tables: ‘nit\_la\_fct\_lc\_<SMO\_NAME>’,’ nit\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME>’ and ‘nit\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ and includes rows where ‘NIT\_AMT’ is NOT NULL and GEO\_ID value specific to SMO.
* At last table ‘nit\_la\_fct\_join\_curr\_<SMO\_NAME>’ is created by performing left join between the table ‘nit\_la\_fct\_lc\_<SMO\_NAME>’ and the tables ‘nit\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME>’ , ‘nit\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ and ‘rls\_security\_v\_<SMO\_NAME>’ to join different currency types and add rls\_key.

### 10.3.6 Processing SD data(650\_LA\_SD\_DIM\_FACT\_PROCESS\_<SMO\_NAME>):

* Table created repo\_la.sd\_hier\_dim\_ARGENTINA
* hard coded values inserted as below

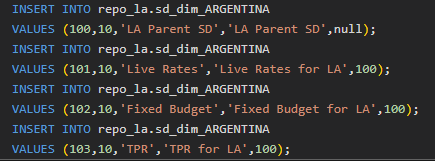
SD\_HIER\_ID=’10’

SD\_HIER\_NAME=’ SD\_LA’

SD\_HIER\_DESCRIPTION=’ Base SD Hierarchy for LA region’

SD\_HIER\_OWNER=’ CNGC’

* Smilaary repo\_la.sd\_dim\_ARGENTINA table is created.
* hard coded values inserted as below:



* ‘sd\_fixed\_budget\_la<SMO\_NAME>table created from ‘sd\_fixed\_budget\_la \_v\_<SMO\_NAME>’ and ‘manual\_customer\_mapping\_v\_<SMO\_NAME>’ tables and joins them with ‘mnr\_val\_fund\_opt\_dim\_vw\_v' and ‘mnr\_fund\_sbstr\_mapng\_dim\_vw\_v’ tables. This table has the values for SD fixed budgets for LA.
* ‘sd\_tpr\_la\_v\_<SMO\_NAME>’ view is created from table ‘mnr\_sd\_tpr\_dim\_vw\_v’ by joining on ‘MNR\_LE\_MAPNG\_DIM\_MV’ and ‘vbrp\_v\_ARGENTINA’ by only selecting <SMO\_NAME>.This table has the SD\_TPR values which are receiving from SAP data.
* ‘sd\_live\_rates\_la\_<SMO\_NAME>’ table created from ‘manual\_customer\_mapping\_v\_<SMO\_NAME>’ by joining sd\_live\_rates\_la\_v\_<SMO\_NAME> view and ‘mnr\_geo\_mapng\_dim\_vw\_unified\_v’ rounding out the data depending on fund\_id\_rnum value, if fund\_id\_rnum=-1 Then sd\_tpr, fund\_id\_rnum=-2 then sd\_fixed\_budget and fund\_id\_rnum=0/1/2 then sd\_live\_rates grouping on reporting\_customer\_id,month, product, currency granularity.
* ‘sd\_shp\_alloc\_live\_rates\_la\_<SMO\_NAME>’ table is created which allocated sm\_amt coming from different sources from ‘dir\_ship\_la\_fct\_<SMO\_NAME>’ joining with ‘sd\_live\_rates\_la\_<SMO\_NAME>’, with union of all SD\_AMT on difference sources like SECTOR, SUBSECTOR, BRAND, CATEGORY AND COMPANY. The sd\_amt is calculated using formula
* ROUND((ROUND(GIV\_AMT,8)/ROUND(value,8)) \* sd\_live\_rates,8) SD\_AMT, Where value is depends on the SD source column.Also partitioning GIV on SECTOR, SUBSECTOR,BRAND,CATEGORY AND COMPANY.
* ‘sd\_shp\_alloc\_fixed\_budget\_la\_<SMO\_NAME>’ is created which allocated sd fixed budget values. Table is created from dir\_ship\_la\_fct\_<SMO\_NAME> BY JOINING with ‘sd\_fixed\_budget\_la\_<SMO\_NAME> on subsector and month granularity.Also summing giv value by partitioning on reporting\_customer,month and subsector level.
* ‘sd\_shp\_alloc\_tpr\_la\_<SMO\_NAME>’ is created which allocated sd tpr values. Table is created from dir\_ship\_la\_fct\_<SMO\_NAME> BY JOINING with ‘sd\_fixed\_budget\_la\_<SMO\_NAME> on subsector and month granularity.Also summing giv value by partitioning on reporting\_customer,month and subsector level.
* ‘sd\_la\_fct\_lc\_<SMO\_NAME>’ table is created from union on sd\_shp\_alloc\_live\_rates\_<SMO\_NAME>, sd\_shp\_alloc\_fixed\_budget\_<SMO\_NAME>,
* sd\_shp\_alloc\_tpr\_<SMO\_NAME>
* ‘repo\_la.sd\_la\_fct\_<SMO\_NAME>’table is created by union of local and reporting currency sd\_la\_fct\_rusd\_<SMO\_NAME> and sd\_la\_fct\_cusd\_<SMO\_NAME> tables y joining with ‘rls\_security\_v\_<SMO\_NAME>’ required for row level security and converts currency from original currency to USD using the exchange rates from ‘frgn\_exchg\_rate\_dim\_v’ table.
* ‘nit\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ retrieves data from ‘nit\_la\_fct\_lc\_<SMO\_NAME>’ and joins with table ‘FX\_mth\_multiple\_currencies\_avg\_closed\_fy\_v’ on the condition that trgt\_iso\_crncy\_code=’USD’ and srce\_iso\_crncy\_code is equal to CURRENCY\_ID of ‘nit\_la\_fct\_lc\_<SMO\_NAME>’.The GIV\_AMT and NIV\_AMT are calculated by multiplying GIV\_AMT and NIV\_AMT values with corresponding exchange rates.
* ‘nit\_la\_fct\_<SMO\_NAME>’ created by selecting data from three different tables: ‘nit\_la\_fct\_lc\_<SMO\_NAME>’,’nit\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME>’ and ‘nit\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ and includes rows where ‘NIT\_AMT’ is NOT NULL and GEO\_ID value specific to SMO.
* At last table ‘nit\_la\_fct\_join\_curr\_<SMO\_NAME>’ is created by performing left join between the table ‘nit\_la\_fct\_lc\_<SMO\_NAME>’ and the tables ‘nit\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME>’ , ‘nit\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ and ‘rls\_security\_v\_<SMO\_NAME>’ to join different currency types and add rls\_key.

### 10.3.7 Allocation of NSRD and GS(750\_LA\_NSRD\_GS\_FACT\_PROCESS\_<SMO\_NAME>)

* ‘nsrd\_gross\_sales\_la\_v\_<SMO\_NAME>’ View is created from glpca\_v\_<SMO\_NAME> with joining ‘mnr\_sap\_shipt\_mapng\_dim\_vw\_v’ , ‘mnr\_geo\_mapng\_dim\_vw\_v’,

’ gross\_saless\_account\_list\_v\_<SMO\_NAME>’, ‘SODE\_PC084\_FDIM’ . this table has the NSRD values which would require for performing the Walmart and pre Walmart allocation.

PreWalmart logic:

* The main assumption of this step is the reallocation of GS / NSRD values from customers who do not have Shipments to those who do.
* The logic is as follows : Based on the shipment source create all artificial records where to reallocate all transferrable values from SAP source. Merge all artificial records with source SAP data to produce a new SAP source table for the rest 750 notebook calculations.
* The PreWalmart algorithm implementation consists of 6 steps:

1. SAP Source data enrichment and designation of records for value reallocation
2. Shipment data prepare
3. Creation of artificial records with new, reallocated values
4. Flag transferred SAP source records
5. Merge of artificial records and not transferred SAP data
6. New replacement for SAP source data creation

* Step 1 – SAP Source data enrichment and designation of records for value reallocation

1. ‘pre\_walmart\_10\_<SMO\_NAME>’ table is created from nsrd\_gross\_sales\_la\_v\_<SMO\_NAME> joining on’SODE\_CAL\_FDIM’ and ‘manual\_customer\_mapping\_v\_<SMO\_NAME>’required for Reporting Customer partitioned by TIME\_ID to prepare SAP data to reallocation process and to enrich them with some flags and additional attributes like REPORTING\_CUSTOMER\_ID, TIME\_ID, FLG\_DATA\_PATTERN which are necessary to perform join with Shipments source and implement PreWalmart logic.
2. stage\_la\_excels.manual\_customer\_mapping\_v\_\_<SMO\_NAME> view to get REPORTING\_CUSTOMER\_ID and repo\_la.dir\_ship\_la\_fct\_lc to get information about Shipment existence (FLG\_SHIPMENT).
3. Additionally stage\_common.SODE\_CAL\_FDIM use for time dimension table to convert SAP Fiscal Year information to the standard TIME\_ID attribute necessary for join with Shipments data.
4. stage\_mmr.mnr\_pc\_sbstr\_nsrd\_dim\_vw use to get Sub Sector attribute, which is used to calculate the DATA\_PATTERN flag. As additional enrichment process, we are calculating here ROWID attribute, which is unique record ID necessary to simplify the process of flagging those records where we have reallocated their values.

* Step 2 - Shipment data prepare.

1. In this step we are creating a stage\_work\_la.pre\_walmart\_shipment DELTA table, which is aggregated and narrowed to optimize the join with SAP data.
2. We are using windowing functions here to calculate GIV totals on 3 different levels (from the lowest to the highest order):
3. GIV\_FPC – most detailed GIV calculation based on FPC level (partition by reporting\_customer\_id, geo\_id, customer\_id, time\_id, currency\_id, proft\_ctr\_3\_id, fpc)
4. GIV\_PC – total GIV calculation on Profit Center level (PC\_lv\_3) (partition by reporting\_customer\_id, geo\_id, time\_id, currency\_id, proft\_ctr\_3\_id)
5. GIV\_RC – total GIV calculation on Reporting Customer level (partition by reporting\_customer\_id, geo\_id, time\_id, currency\_id)

* Step 3 – Creation of artificial records with new, reallocated values

1. In this step we are creating a stage\_work\_la.pre\_walmart\_20 DELTA table partitioned by TIME\_ID to calculate artificial SAP records with new, reallocated values.
2. The calculation is based on both previous tables stage\_work\_la.pre\_walmart\_10 (pw) and stage\_work\_la.pre\_walmart\_shipment\_<SMO\_NAME> (sh). We are joining them based on GEO\_ID, TIME\_ID, CURRENCY\_ID, REPORTING\_CUSTOMER\_ID, and on the special condition for the PROFT\_CTR\_3\_ID column:
3. sh.proft\_ctr\_3\_id = case when PROFIT\_CTR\_ID is not null then pw.proft\_ctr\_3\_id else sh.proft\_ctr\_3\_id end This is because when there is no Profit Center level 3 on the SAP side, we need to join data on the higher, Reporting Customer level.
4. This has also impact on the GIV\_FACTOR calculation. For the join on the Profit Center level we are calculating it as GIV\_FPC/ GIV\_PC and for the Reporting Customer level join it is calculated as GIV\_FPC / GIV\_RC.
5. This new table gets all the stage\_work\_la.pre\_walmart\_10 attributes (columns) that later will be used to the merge back with this source table.

* Step 4 – Flag transferred SAP source records

1. In this step we are using previously calculated ROWID column to update stage\_work\_la.pre\_walmart\_10\_<SMO\_NAME> records with flag that specific record value was reallocated. We also nullify value of such source record. To do that we are using stage\_work\_la.pre\_walmart\_20\_<SMO\_NAME> table ROWID values taken from source records used to create those artificial records.

* Step 5 – Merge of artificial records and not transferred SAP data

1. In this step we are merging “not transferrable” records of stage\_work\_la.pre\_walmart\_10\_<SMO\_NAME> to the stage\_work\_la.pre\_walmart\_20\_<SMO\_NAME> table. We are doing that on the most detailed level of the both those tables.

* Step 6 – New replacement for SAP source data creation

1. In this last step we are creating a new table stage\_work\_la.nsrd\_gs\_pre\_walmart\_<SMO\_NAME> that will replace stage\_la\_simplement.nsrd\_gross\_sales\_v\_<SMO\_NAME> in the rest of 750 notebook logic (in creation scripts of tables:

1.stage\_work\_la.ship\_step\_10\_sap\_gs\_nsrds\_work\_<SMO\_NAME>, 2.stage\_work\_la.ship\_step\_20\_sap\_gs\_nsrds\_cust\_mth\_profit\_work\_<SMO\_NAME>, 3.stage\_work\_la.ship\_step\_30\_sap\_gs\_nsrds\_cust\_mth\_profit\_nodefault\_work\_<SMO\_NAME>)

1. After this step

repo\_la.nsrd\_gs\_la\_fct\_lc\_shp\_exist\_10\_<SMO\_NAME>

repo\_la.nsrd\_gs\_la\_fct\_lc\_shp\_exist\_20\_<SMO\_NAME>

repo\_la.nsrd\_gs\_la\_fct\_lc\_shp\_exist\_30\_<SMO\_NAME>

tables are created from the scripts table created in above step.

Which are used to calculate the NSRD and GS pre-Walmart scenarios in the next subsequent step ’ nsrd\_gs\_la\_fct\_lc\_giv0\_<SMO\_NAME>.

Next step in 750 notebook is to create

‘nsrd\_gs\_la\_fct\_lc\_prod\_exclud\_<SMO\_NAME>’ table from ‘nsrd\_gs\_la\_fct\_lc\_<SMO\_NAME>’ by joining on geo, product mapping and prod hierarchy tables. Which filtering out products that are part of exclusion list and provides data for provides data for Walmart scenario.

WALMART SCENARIO:

Aggregate the derived GS values to Reported Customer Level & allocate it down to Cust L9s based on % GIV. The same approach to be followed for NSRds (From SAP) as the source table for both Gross Sales & NSRd is the same.

nsrd\_gs\_la\_fct\_lc\_allocated\_<SMO\_NAME> table is created

where allocation of NSRD is performed using the approach explained above and the formula used is as follows

CASE WHEN CAST(fct.TOTAL\_REPORTING\_GIV AS DECIMAL) <> 0.00

THEN ROUND(fct.TOTAL\_NSRD\_DAMAGE\_RETURNS\_AMT \* (ROUND(f.GIV\_AMT,8)/ROUND(fct.TOTAL\_REPORTING\_GIV,8)),8)

ELSE NSRD\_DAMAGE\_RETURNS\_AMT

END NSRD\_DAMAGE\_RETURNS\_AMT\_NEW

By joining on prod hierarchy.

‘nsrd\_gs\_la\_fct\_lc\_allocated\_mex\_allocation\_<SMO\_NAME>table consist of main allocation of nsrd and gs which uses the formula

round(ab.NSRD\_PROMPT\_PAYMENT\_AMT\_NEW \* (round(s.GIV\_AMT,8)/(round(r.TOTAL\_GIV\_WITHOUT\_PHC,8))),8) NEWLY\_ALLOCATED\_NSRD\_PP,

form table ’nsrd\_gs\_la\_fct\_lc\_allocated\_mex\_exclusion\_base\_<SMO\_NAME>’

by joining

1.’ nsrd\_gs\_la\_fct\_lc\_allocated\_mex\_exclusion\_aggregated\_<SMO\_NAME>’

2.’ SODE\_P5005\_FDIM’

3.’ nsrd\_gs\_la\_fct\_lc\_allocated\_giv\_rest\_phc\_<SMO\_NAME>’

4.’ nsrd\_gs\_la\_fct\_lc\_allocated\_mex\_exclusion\_aggregated\_base\_<SMO\_NAME>’

The allocation logic allocates the NSRD in the cases where GIV is not equal to Zero in final derived GS & NSRDs based on existing design – The values to be aggregated to Reporting Customer – Month level and allocating down to Cust L9 –FPC - Month level based on % GIV.

### 10.3.8 Allocation of Customer manual inputs(790\_LA\_MANUAL\_CUSTOMER\_INPUT\_ALLOCATION\_<SMO\_NAME>):

* **‘customer\_manual\_input\_alloc\_la\_fct\_lc\_<SMO\_NAME>’** iscreated by selecting ‘GIV\_AMT’ from ‘dir\_ship\_la\_fct**\_<SMO\_NAME>**’ and joining with ‘manul\_customer\_input\_v**\_<SMO\_NAME>**’ from which following columns ‘NSRD\_MANUL\_INPUT\_LC\_VAL’ and ‘SD\_MANUL\_INPUT\_LC\_VAL’ are selected for calculation and following filters are applied to resultant data: NSRD\_MANUL\_INPUT\_LC\_VAL IS NOT NULL OR SD\_MANUL\_INPUT\_LC\_VAL IS NOT NULL AND GEO\_ID='SMO SPECIFIC GEO\_ID' .
* ’NSRD\_MANUAL\_INPUT’ is calculated using the following expression: (GIV\_AMT/SUM\_GEO\_CAT\_MTH\_PC\_CUST4\* NSRD\_MANUL\_INPUT\_LC\_VAL)
* ‘SD\_MANUAL\_INPUT’ is calculated using the following expression: (GIV\_AMT/SUM\_GEO\_CAT\_MTH\_PC\_CUST4\* SD\_MANUL\_INPUT\_LC\_VAL)
* **‘manual\_cust\_input\_step\_10\_<SMO\_NAME>’** created by selecting columns from ‘dir\_ship\_la\_fct**\_<SMO\_NAME>**’ which are used to calculate ‘SUM\_GEO\_PC\_CAT\_MTH\_CUST9 ’ and ‘SUM\_GEO\_PC\_CAT\_FPC\_CUST4\_FY’ and ‘GIV\_ALLOCATION\_FACTOR’ by doing partitioning of SUM(GIV\_AMT) by category level and applying filters on the resultant data as: ’GIV\_AMT’ is not null or zero and GEO\_ID specific to SMO.
* **‘manual\_cust\_input\_step\_20\_<SMO\_NAME>’** is created by selecting data from ‘**dir\_ship\_la\_fct\_<SMO\_NAME>**’. The ‘SUM\_GEO\_PC\_CAT\_MTH\_CUST9’ and ‘SUM\_GEO\_PC\_CAT\_FPC\_CUST4\_FY’ is calculated using sum(GIV\_AMT) partitioning it by sub-sector level and filters applied as: GIV\_AMT is not null or zero and GEO\_ID specific to SMO.
* The table **‘manual\_cust\_input\_step\_30\_<SMO\_NAME>’** selects data from two tables ‘dir\_ship\_la\_fct\_<SMO\_NAME>’ and **‘manul\_customer\_input\_v\_<SMO\_NAME>’** and resultant table is left joined with ‘manual\_cust\_input\_step\_10\_ARGENTINA’ to get ‘NSRD\_MANUAL\_INPUT’ and ‘SD\_MANUAL\_INPUT’ kpi’s.
* ‘manual\_cust\_input\_step\_40\_<SMO\_NAME>’ table is created from repo\_la.manual\_cust\_input\_step\_30\_ARGENTINA and by joining with repo\_la.manual\_cust\_input\_step\_20\_ARGENTINA Allocation NSRD\_MANUAL\_INPUT Values as NSRD\_MANUL\_INPUT\_LC\_VAL \* x.GIV\_ALLOCATION\_FACTOR\_STEP20 And SD\_MANUAL\_INPUT As SD\_MANUL\_INPUT\_LC\_VAL \* x.GIV\_ALLOCATION\_FACTOR\_STEP20 to look for values that are still unallocated after we allocated values on CATEGORY level.
* To prevent double allocation of data, the sub sector level allocation process will be conducted using the table from ‘manual\_cust\_input\_step\_30\_<SMO\_NAME>’ and table repo\_la.customer\_manual\_input\_alloc\_la\_fct\_lc\_<SMO\_NAME> new table repo\_la.fmr\_tie\_out\_la\_fct\_lc\_customer\_manual\_input\_allocated\_<SMO\_NAME> is created.

### 10.3.9 Calculation of Tie-Out's(800\_LA\_FMR\_TIE\_OUT\_FACT\_PROCESS\_<SMO\_NAME>):

KPIs involved in calculation of Tieouts – FMR Inputs, All other inputs related to NSRd & SD.

As FMR Inputs are provided at Category level in Uploader, the Tieouts are calculated at Category level.

* FMR Inputs provided in $ rate is converted to LC using periodic FX Rate
* All other KPIs are aggregated at category level
* Formula is then processed at Category level
* The value obtained for a given category is aggregated down to FPC level, based on %GIV value of that FPC for that given category.
* If, give category doesn’t have shipments for that month, then Tie Outs is calculated using the shipments pattern (CFY & PFY months) of that specific category. (No Sales scenario of Tieouts)
* table stage\_work\_la.ship\_step\_10\_fmr\_input\_la\_aggr\_v\_<SMO\_NAME> is created using stage\_mmr.mnr\_fm\_rptng\_input\_dim\_vw\_v and join with LEFT OUTER JOIN stage\_mmr.mnr\_geo\_mapng\_dim\_vw\_v
* VIEW stage\_work\_la.ship\_step\_10\_fmr\_input\_aggr\_v\_<SMO\_NAME> is created from view stage\_work\_la.ship\_step\_10\_fmr\_input\_la\_aggr\_v\_<SMO\_NAME>
* table repo\_la.missing\_data\_nit\_<SMO\_NAME> is created using repo\_la.nit\_la\_fct\_<SMO\_NAME> selected the rows which are excluded inrepo\_la.dir\_ship\_la\_fct\_<SMO\_NAME> and by join with LEFT OUTER JOIN stage\_common.SODE\_CAL\_FDI and LEFT JOIN stage\_common.mnr\_prod\_hier\_mapng\_dim\_vw\_v
* Filter applied PROD\_HIER\_ID = 5005
* Table stage\_work\_la.ship\_coeff\_aggr\_v\_<SMO\_NAME> is created using tbalerepo\_la.dir\_ship\_la\_fct\_<SMO\_NAME> union with repo\_la.missing\_data\_nit\_<SMO\_NAME> and by joining with LEFT OUTER JOIN repo\_la.nit\_la\_fct\_<SMO\_NAME>,LEFT OUTER JOIN repo\_la.nsrd\_gs\_la\_fct\_<SMO\_NAME>, LEFT OUTER JOIN repo\_la.tdc\_la\_fct\_<SMO\_NAME> and LEFT OUTER JOIN stage\_work\_la.sd\_optima\_total\_sum\_work\_v\_<SMO\_NAME>
* New table stage\_work\_la.ship\_fmr\_tie\_out\_v\_<SMO\_NAME> is created. As we are combining values from multiple sources, we are using COALESCE function here to fill up any gaps that might exist.repo\_la.fmr\_tie\_out\_la\_fct\_lc\_customer\_manual\_input\_allocated\_<SMO\_NAME> is used here to provide values from notebook 790.
* New Table ship\_fmr\_tie\_out\_alloc\_la\_fct\_lc\_<SMO\_NAME> is created. Containing the case statement formula to calculate TIE\_OUTS As (giv/sum(giv) over partition on geo, category, time, and profit centre)/tieout in local currency calculated in above step.
* New table ship\_fmr\_tie\_out\_step\_10\_<SMO\_NAME> is created from dir\_ship\_la\_fct\_<SMO\_NAME>which portioned the giv over geo, profit centre,
* Product and customer\_level for next subsequent steps.
* New table stage\_work\_la.fmr\_tie\_out\_la\_fct\_lc\_allocated\_<SMO\_NAME> created. It is containing the data from stage\_work\_la.ship\_fmr\_tie\_out\_alloc\_la\_fct\_lc\_<SMO\_NAME> and stage\_work\_la.ship\_fmr\_tie\_out\_step\_30\_<SMO\_NAME> with additional filter of product\_id IS NOT NULL to exclude records without any shipments in the last 2 fiscal years. This table contains data that is already allocated.
* New Table fmr\_tie\_out\_la\_fct\_lc\_final\_<SMO\_NAME> is created from fmr\_tie\_out\_la\_fct\_lc\_allocated\_<SMO\_NAME> COALESCE function is used to have the lpi’s value from either tie out or allocated values.

### 10.3.10 Processing of Gross Contribution(850\_LA\_GROSS\_CONTRIBUTION\_FACT\_PROCESS\_<SMO\_NAME>)

* Table ‘gross\_contribution\_la\_fct\_lc\_<SMO\_NAME>’ is created by using repo\_la.dir\_ship\_la\_fct\_<SMO\_NAME> and by joining with LEFT OUTER JOIN   repo\_la.nsrd\_gs\_la\_fct\_<SMO\_NAME> LEFT OUTER JOIN stage\_work\_la.sd\_optima\_total\_sum\_work\_v\_<SMO\_NAME> FULL OUTER JOIN repo\_la.fmr\_tie\_out\_la\_fct\_lc\_final\_<SMO\_NAME>  LEFT OUTER JOIN repo\_la.tdc\_la\_fct\_<SMO\_NAME> LEFT OUTER JOIN repo\_la.nit\_la\_fct\_<SMO\_NAME> has calculated measures such as: SAP\_GROSS\_SALES\_AMT,TIE\_OUT\_NSRD,TIE\_OUT\_SD,TIE\_OUT\_TDC,NSRD\_TOTAL,SUM\_SD\_AMT,SD\_TOTAL,TDC\_AMT,NOS\_LC,GC\_LC
* GC\_LC value is calculated using the formula: GC\_LC = NOS\_LC - TDC\_AMT - TIE\_OUT\_TDC
* NOS\_LC value is calculated using the formula: NOS\_LC = SAP\_GROSS\_SALES\_AMT - NSRD\_TOTAL - SD\_TOTAL
* NSRD\_TOTAL is calculated using the formula: NSRD\_TOTAL = nsrd\_damage\_returns\_amt + nsrd\_price\_adjustments\_amt + nsrd\_distributor\_discounts\_amt + nsrd\_others\_amt + nsrd\_prompt\_payment\_amt + nsrd\_logistic\_discounts\_amt + nsrd\_write\_off\_amt + nsrd\_unknown\_amt + NSRD\_MANUAL\_INPUT + TIE\_OUT\_NSRD + NIT\_AMT
* SD\_TOTAL is calculated using the formula: SD\_TOTAL = SUM\_SD\_AMT + SD\_MANUAL\_INPUT + TIE\_OUT\_SD
* Created gross\_contribution\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME> by suing repo\_la.gross\_contribution\_la\_fct\_lc\_<smo\_name> and joing with stage\_common.frgn\_exchg\_rate\_dim\_v for reporting dollar conversion by multiplying measures with reporting dollar value along with multiplier value for local currency.
* Created gross\_contribution\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME> by suing repo\_la.gross\_contribution\_la\_fct\_lc\_<SMO\_NAME> and jonning with LEFT OUTER JOIN stage\_common.FX\_mth\_multiple\_currencies\_avg\_closed\_fy\_v multiplying measures with constant dollar value.
* Table ‘gross\_contribution\_la\_fct\_join\_curr\_<SMO\_NAME>’ is created by performing left join between the table ‘gross\_contribution\_la\_fct\_lc\_<SMO\_NAME>’ and the tables ‘gross\_contribution\_la\_fct\_lc\_rusd\_v\_<SMO\_NAME>’ , ‘gross\_contribution\_la\_fct\_lc\_cusd\_v\_<SMO\_NAME>’ and ‘rls\_security\_v\_<SMO\_NAME>’ to join different currency types and add rls\_key.

### 10.3.11 Creating final flat output table(890\_LA\_FLAT\_FCT\_<SMO\_NAME>)

* A new table la\_flat\_output\_fct\_<SMO\_NAME> has been created to consolidate all final output KPIs using timeid as filter.

|  |  |
| --- | --- |
| KPI | TABLE NAME |
| GIV\_AMT\_LC, NIV\_AMT, VOLUME | dir\_ship\_la\_fct\_join\_curr\_<SMO\_NAME> |
| TDC\_AMT\_LC | repo\_la.tdc\_la\_fct\_join\_curr\_\_<SMO\_NAME> |
| NIT\_AMT\_LC, | nit\_la\_fct\_join\_curr\_\_<SMO\_NAME> |
| NSRD\_DAMAGE\_RETURNS\_AMT\_LC, NSRD\_PRICE\_ADJUSTMENTS\_AMT\_LC, NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_LC, NSRD\_OTHERS\_AMT\_LC, NSRD\_PROMPT\_PAYMENT\_AMT\_LC, NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_LC, NSRD\_WRITE\_OFF\_AMT\_LC, SAP\_GROSS\_SALES\_AMT\_LC | nsrd\_gs\_la\_fct\_join\_curr\_<SMO\_NAME> |
| CAT\_NSRD\_TIE\_OUT\_LC,  CAT\_SD\_TIE\_OUT\_LC, CAT\_TDC\_TIE\_OUT\_LC,  TIE\_OUT\_NSRD\_LC,  TIE\_OUT\_SD\_LC,  TIE\_OUT\_TDC\_LC,  NSRD\_MANUAL\_INPUT\_LC | fmr\_tie\_out\_la\_fct\_join\_curr\_<SMO\_NAME> |
| SAP\_GROSS\_SALES\_AMT,  TIE\_OUT\_NSRD,  TIE\_OUT\_SD,  TIE\_OUT\_TDC,  TDC\_AMT,  SD\_TOTAL\_LC,  NSRD\_TOTAL\_LC,  NOS\_LC, GC\_LC | gross\_contribution\_la\_fct\_join\_curr\_<SMO\_NAME> |

* A new table la\_sd\_flat\_output\_fct\_<SMO\_NAME> has been created to consolidate all final SD output KPIs using timeid as filter.

|  |  |
| --- | --- |
| KPI | TABLE NAME |
| SD\_LIVE\_RATES\_AMT\_LC, SD\_FIXED\_BUDGET\_AMT\_LC, SD\_TPR\_AMT\_LC | sd\_la\_fct\_join\_cur\_<SMO\_NAME> |
| SD\_MANUAL\_INPUT\_LC | fmr\_tie\_out\_la\_fct\_join\_curr\_<SMO\_NAME> |

## OUTPUT DIMS AND FACTS

### 10.4.1 01\_FINAL\_LA\_LADMAR\_OUTPUT

* Created repo\_la.la\_flat\_output\_fct to consolidate data from 7 existing tables:

repo\_la.la\_flat\_output\_fct\_ARGENTINA,

repo\_la.la\_flat\_output\_fct\_BRAZIL,

repo\_la.la\_flat\_output\_fct\_CHILE,

repo\_la.la\_flat\_output\_fct\_COLOMBIA,

repo\_la.la\_flat\_output\_fct\_MEXICO,

repo\_la.la\_flat\_output\_fct\_PERU,

repo\_la.la\_flat\_output\_fct\_LADMAR

* Created repo\_la.la\_sd\_flat\_output\_fct to consolidate data from 7 existing tables:

repo\_la.la\_ sd\_flat\_output\_fct\_ARGENTINA,

repo\_la.la\_ sd\_flat\_output\_fct\_BRAZIL,

repo\_la.la\_ sd\_flat\_output\_fct\_CHILE,

repo\_la.la\_ sd\_flat\_output\_fct\_COLOMBIA,

repo\_la.la\_ sd\_flat\_output\_fct\_MEXICO,

repo\_la.la\_ sd\_flat\_output\_fct\_PERU,

repo\_la.la\_ sd\_flat\_output\_fct\_LADMAR

* Union All combines the rows from all 7 tables into a single result set without removing any duplicates.
* Created stage\_la\_excels.rls\_security to consolidate data from 7 existing tables into one from above mentioned SMO specific table

Which includes VIWS:

stage\_la\_excels.rls\_security\_v\_ARGENTINA

stage\_la\_excels.rls\_security\_v\_BRAZIL

stage\_la\_excels.rls\_security\_v\_CHILE

stage\_la\_excels.rls\_security\_v\_COLOMBIA

stage\_la\_excels.rls\_security\_v\_MEXICO

stage\_la\_excels.rls\_security\_v\_PERU

stage\_la\_excels.rls\_security\_v\_ladmar

* Created stage\_la\_fix\_map.geo\_56\_mapping to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes table:

stage\_la\_fix\_map.geo\_56\_mapping\_ARGENTINA

stage\_la\_fix\_map.geo\_56\_mapping \_BRAZIL

stage\_la\_fix\_map.geo\_56\_mapping \_CHILE

stage\_la\_fix\_map.geo\_56\_mapping \_COLOMBIA

stage\_la\_fix\_map.geo\_56\_mapping \_MEXICO

stage\_la\_fix\_map.geo\_56\_mapping \_PERU

stage\_la\_fix\_map.geo\_56\_mapping \_ladmar

* Created repo\_la.dir\_ship\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.dir\_ship\_la\_fct\_ARGENTINA

repo\_la.dir\_ship\_la\_fct\_BRAZIL

repo\_la.dir\_ship\_la\_fct\_CHILE

repo\_la.dir\_ship\_la\_fct\_COLOMBIA

repo\_la.dir\_ship\_la\_fct\_MEXICO

repo\_la.dir\_ship\_la\_fct\_PERU

repo\_la.dir\_ship\_la\_fct\_LADMAR

* Created repo\_la.tdc\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.tdc\_la\_fct \_ARGENTINA

repo\_la.tdc\_la\_fct \_BRAZIL

repo\_la.tdc\_la\_fct \_CHILE

repo\_la.tdc\_la\_fct \_COLOMBIA

repo\_la.tdc\_la\_fct \_MEXICO

repo\_la.tdc\_la\_fct \_PERU

repo\_la.tdc\_la\_fct \_LADMAR

* Created repo\_la.nit\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.nit\_la\_fct\_ARGENTINA

repo\_la.nit\_la\_fct\_BRAZIL

repo\_la.nit\_la\_fct\_CHILE

repo\_la.nit\_la\_fct\_COLOMBIA

repo\_la.nit\_la\_fct\_MEXICO

repo\_la.nit\_la\_fct\_PERU

repo\_la.nit\_la\_fct\_LADMAR

* Created repo\_la.nsrd\_gs\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.nsrd\_gs\_la\_fct \_ARGENTINA

repo\_la.nsrd\_gs\_la\_fct \_BRAZIL

repo\_la.nsrd\_gs\_la\_fct \_CHILE

repo\_la.nsrd\_gs\_la\_fct \_COLOMBIA

repo\_la.nsrd\_gs\_la\_fct \_MEXICO

repo\_la.nsrd\_gs\_la\_fct \_PERU

repo\_la.nsrd\_gs\_la\_fct \_LADMAR

* Created repo\_la.fmr\_tie\_out\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.fmr\_tie\_out\_la\_fct \_ARGENTINA

repo\_la.fmr\_tie\_out\_la\_fct \_BRAZIL

repo\_la.fmr\_tie\_out\_la\_fct \_CHILE

repo\_la.fmr\_tie\_out\_la\_fct \_COLOMBIA

repo\_la.fmr\_tie\_out\_la\_fct \_MEXICO

repo\_la.fmr\_tie\_out\_la\_fct \_PERU

repo\_la.fmr\_tie\_out\_la\_fct \_LADMAR

* Created repo\_la.gross\_contribution\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.gross\_contribution\_la\_fct \_ARGENTINA

repo\_la.gross\_contribution\_la\_fct \_BRAZIL

repo\_la.gross\_contribution\_la\_fct \_CHILE

repo\_la.gross\_contribution\_la\_fct \_COLOMBIA

repo\_la.gross\_contribution\_la\_fct \_MEXICO

repo\_la.gross\_contribution\_la\_fct \_PERU

repo\_la.gross\_contribution\_la\_fct \_LADMAR

* Created repo\_la.sd\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables

Which includes tables:

repo\_la.sd\_la\_fct \_ARGENTINA

repo\_la.sd\_la\_fct \_BRAZIL

repo\_la.sd\_la\_fct \_CHILE

repo\_la.sd\_la\_fct \_COLOMBIA

repo\_la.sd\_la\_fct \_MEXICO

repo\_la.sd\_la\_fct \_PERU

repo\_la.sd\_la\_fct \_LADMAR

* Created repo\_la.sd\_dim to consolidate data from 7 existing tables into one from above mentioned SMO specific tables.

Which includes tables:

repo\_la.sd\_dim \_ARGENTINA

repo\_la.sd\_dim \_BRAZIL

repo\_la.sd\_dim \_CHILE

repo\_la.sd\_dim \_COLOMBIA

repo\_la.sd\_dim\_MEXICO

repo\_la.sd\_dim \_PERU

repo\_la.sd\_dim\_LADMAR

* Created repo\_la.pea\_la\_fct to consolidate data from 7 existing tables into one from above mentioned SMO specific tables.

Which includes tables:

repo\_la.pea\_la\_fct \_ARGENTINA

repo\_la.pea\_la\_fct \_BRAZIL

repo\_la.pea\_la\_fct \_CHILE

repo\_la.pea\_la\_fct \_COLOMBIA

repo\_la.pea\_la\_fct \_MEXICO

repo\_la.pea\_la\_fct \_PERU

repo\_la.pea\_la\_fct \_LADMAR

### 10.4.2 Creating output with RLS security(02\_900\_LA\_OUTPUT)

* Created repo\_la\_output.rls\_security and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/rls\_security' by using stage\_la\_excels.rls\_security\_v
* Created repo\_la\_output.la\_flat\_output\_fct and loaded as parquet file in below location

path='dbfs:/mnt/cngc-storage/dim\_fct\_output/la\_flat\_output\_fct' by using repo\_la.la\_flat\_output\_fct with filters:

TIME\_ID < CONCAT(YEAR(CURRENT\_DATE), LPAD(MONTH(CURRENT\_DATE),2,'0'))

AND GEO\_ID IN (select distinct GEO\_ID from tech.config\_output\_geo\_ids\_4\_aas\_v con where REGION = 'LA')

or GEO\_ID IN (select distinct GEO\_MAP\_ID from stage\_la\_fix\_map.geo\_56\_mapping con))

* Created repo\_la\_output.la\_sd\_flat\_output\_fct and loaded as parquet file in below location

path='dbfs:/mnt/cngc-storage/dim\_fct\_output/la\_sd\_flat\_output\_fct' by using repo\_la.la\_sd\_flat\_output\_fct with same below mentioned filter:

 TIME\_ID < CONCAT(YEAR(CURRENT\_DATE), LPAD(MONTH(CURRENT\_DATE),2,'0'))

AND GEO\_ID IN (select distinct GEO\_ID from tech.config\_output\_geo\_ids\_4\_aas\_v con where REGION = 'LA')

 or GEO\_ID IN (select distinct GEO\_MAP\_ID from stage\_la\_fix\_map.geo\_56\_mapping con))

### 10.4.3 Creation of filtered out Dims and fact output( 03\_930\_FILTERED\_DIMS\_OUTPUT)

* Created repo\_la\_output.sode\_cal\_time\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_cal\_time\_dim' by using tables stage\_common.sode\_cal\_fdim , repo\_la\_output.la\_flat\_output\_fct and filters CAST(MTH\_NUM AS INT) NOT IN

(180001,183710)

AND CAST (MTH\_NUM AS INT) IN

(SELECT TIME\_ID FROM repo\_la\_output.la\_flat\_output\_fct)

* Created repo\_la\_output.sode\_cal\_time\_dim\_rolling and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_cal\_time\_dim\_rolling' by selecting time\_id , time\_id-100 from repo\_la\_output.sode\_cal\_time\_dim
* Created repo\_la\_output.sode\_pc064\_profit\_center\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_pc064\_profit\_center\_dim' by using stage\_common.sode\_pc064\_fdim with filter

PROFT\_CTR\_3\_ID IN (

SELECT PROFIT\_CENTER\_ID FROM repo\_la\_output.la\_flat\_output\_fct )

* Created repo\_la\_output.sode\_g705\_geography\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_g705\_geography\_dim' by joining table stage\_common.sode\_g705\_fdim and stage\_mmr.mnr\_geo\_mapng\_dim\_vw\_v based on GEO\_6\_ID
* Created repo\_la\_output.sode\_c898\_customer\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_c898\_customer\_dim' by using stage\_common.sode\_c898\_fdim based on filter

CUST\_9\_ID, CUST\_ID present in repo\_la\_output.la\_flat\_output\_fct

* Created repo\_la\_output.sode\_p5005\_product\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_p5005\_product\_dim' by using stage\_common.SODE\_P5005\_FDIM based on filter

PROD\_12\_ID present in repo\_la\_output.la\_flat\_output\_fct

* Created repo\_la\_output.sode\_p5801\_product\_dim\_ama and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_p5801\_product\_dim\_ama' by using stage\_common.SODE\_P5801\_FDIM
* Created repo\_la\_output.sode\_p5801\_product\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_p5801\_product\_dim' by using stage\_common.SODE\_P5801\_FDIM
* Created repo\_la\_output.sode\_crncy\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/sode\_crncy\_dim' by using stage\_common.SODE\_CRNCY\_FDIM with condition:

CURRENCY\_NAME NOT IN 'unknownRur' and CURRENCY\_ID present in repo\_la\_output.la\_flat\_output\_fct

* Created repo\_la\_output.manual\_customer\_mapping\_dim and loaded as parquet file in below location path='dbfs:/mnt/cngc-storage/dim\_fct\_output/manual\_customer\_mapping\_dim' by using stage\_common.manual\_customer\_mapping\_dim with condition:

CUSTOMER\_MAPPING\_KEY present in repo\_la\_output.la\_flat\_output\_fct

### 10.4.4 04\_FLAT\_OUTPUT\_MODEL\_EXTRACT

* A table repo\_la.flat\_output\_model\_fct\_dim\_lc is created that contains all the necessary facts and dimension used in Latin America.
* Sd facts which will be further exposed to AAS is sourced from combining two tables repo\_la.sd\_la\_fct and repo\_la.sd\_dim based on sd\_id and filtering sd\_name IN ('Fixed Budget','Live Rates','TPR')
* Remaining final facts and dims are sourced from these respective tables with condition currency\_id NOT IN ('CUSD','RUSD')

|  |  |
| --- | --- |
| MEASURES | TABLE NAME |
| GIV\_AMT\_LC, NIV\_AMT, VOLUME | dir\_ship\_la\_fct\_join\_curr\_<SMO\_NAME> |
| TDC\_AMT\_LC | repo\_la.tdc\_la\_fct\_join\_curr\_\_<SMO\_NAME> |
| NIT\_AMT\_LC, | nit\_la\_fct\_join\_curr\_\_<SMO\_NAME> |
| NSRD\_DAMAGE\_RETURNS\_AMT\_LC, NSRD\_PRICE\_ADJUSTMENTS\_AMT\_LC, NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_LC, NSRD\_OTHERS\_AMT\_LC, NSRD\_PROMPT\_PAYMENT\_AMT\_LC, NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_LC, NSRD\_WRITE\_OFF\_AMT\_LC, SAP\_GROSS\_SALES\_AMT\_LC | nsrd\_gs\_la\_fct\_join\_curr\_<SMO\_NAME> |
| CAT\_NSRD\_TIE\_OUT\_LC,  CAT\_SD\_TIE\_OUT\_LC, CAT\_TDC\_TIE\_OUT\_LC,  TIE\_OUT\_NSRD\_LC,  TIE\_OUT\_SD\_LC,  TIE\_OUT\_TDC\_LC,  NSRD\_MANUAL\_INPUT\_LC | fmr\_tie\_out\_la\_fct\_join\_curr\_<SMO\_NAME> |
| SAP\_GROSS\_SALES\_AMT,  TIE\_OUT\_NSRD,  TIE\_OUT\_SD,  TIE\_OUT\_TDC,  TDC\_AMT,  SD\_TOTAL\_LC,  NSRD\_TOTAL\_LC,  NOS\_LC, GC\_LC | gross\_contribution\_la\_fct\_join\_curr\_<SMO\_NAME> |

* Table repo\_la.flat\_output\_model\_fct\_dim\_rusd is created which includes the data from repo\_la.flat\_output\_model\_fct\_dim\_lc but with an additional filter condition applied: currency\_id IN ('RUSD').
* Table repo\_la.flat\_output\_model\_fct\_dim\_cusd is created which includes the data from repo\_la.flat\_output\_model\_fct\_dim\_lc but with an additional filter condition applied: currency\_id IN ('CUSD').
* Created flat\_output\_model\_fct\_dim\_all to consolidate facts and dimension from three separate tables flat\_output\_model\_fct\_dim\_lc, flat\_output\_model\_fct\_dim\_rusd, flat\_output\_model\_fct\_dim\_cusd
* The table ‘flat\_output\_model\_fct\_dim\_all\_output’ is created by selecting data from table ‘la\_flat\_output\_fct’ and left joining it with filtered dimension tables: ‘sode\_cal\_time\_dim’, ‘sode\_pc064\_profit\_center\_dim’, ‘sode\_g705\_geography\_dim’, ‘sode\_c898\_customer\_dim’, ‘sode\_p5005\_product\_dim’, ‘sode\_crncy\_dim’ and ‘manual\_customer\_mapping\_dim’.
* The table ‘la\_sd\_flat\_output\_fct\_dim\_all\_output’ is created by selecting data from table ‘la\_sd\_flat\_output\_fct’ and left joining it with filtered dimension tables: ‘sode\_cal\_time\_dim’, ‘sode\_pc064\_profit\_center\_dim’, ‘sode\_g705\_geography\_dim’, ‘sode\_c898\_customer\_dim’, ‘sode\_p5005\_product\_dim’, ‘sode\_crncy\_dim’ and ‘manual\_customer\_mapping\_dim’.

### 10.4.5 05\_FLAT\_OUTPUT\_PEA\_MODEL\_EXTRACT

* Created table ‘flat\_output\_pea’ which selects columns ‘CNTRY\_ISO’, ‘PRODUCT\_ID’ from ‘pea\_la\_fct’ and left joining with ‘SODE\_G707\_FDIM’ and ‘mnr\_prod\_hier\_mapng\_dim\_vw\_v’ with filters applied as: prod.PROD\_HIER\_ID = '5005' and prod.CURR\_IND = 'Y' and column ‘PRODUCT\_LVL’ is assigned values based on following case statement conditions:
* WHEN output.FPC\_ID IS NOT NULL THEN 'CU\_GTIN'
* WHEN prod.PROD\_ORIG\_LEVEL\_NUM = 5 THEN 'CATEGORY'
* WHEN prod.PROD\_ORIG\_LEVEL\_NUM = 6 THEN 'BRAND'
* WHEN prod.PROD\_ORIG\_LEVEL\_NUM = 7 THEN 'SEGMENT'
* WHEN prod.PROD\_ORIG\_LEVEL\_NUM = 8 THEN 'BRAND\_FORM'
* ELSE NULL
* Created repo\_la\_output.flat\_output\_pea\_extract and loaded as parquet file in below location path=‘dbfs:/mnt/cngc-storage/pea\_output/la\_pea\_output’ by using table ‘repo\_la.flat\_output\_pea’.

### 10.4.6 PHC output:

* This is a separate kind of output, where we report the output only for PHC(Public Health Care).
* It is generated with 13WD pipeline, which is already coved in ADF section. The logic executed is:
* Table stage\_europe.L9\_PHC\_OUTPUT is created which consists of combined EU and LA data. For LA output source table is repo\_la.la\_flat\_output\_fct joined with repo\_la\_output.sode\_cal\_time\_dim and repo\_la\_output.sode\_p5005\_product\_dim p5005 , stage\_common.frgn\_exchg\_rate\_dim\_v ,stage\_common.FX\_mth\_multiple\_currencies\_avg\_closed\_fy\_v fx\_pfy , repo\_la\_output.sode\_g705\_geography\_dim

Filer applied:

* SUB\_SECTOR\_ID = '1000013898'

## 09\_ARCHIVING

### 10.5.1 01\_ARCHIVING

* Created ‘stage\_la\_excels.manual\_customer\_mapping\_v’ to consolidate data from the following 7 existing tables into one:

stage\_la\_excels.manual\_customer\_mapping\_v\_ARGENTINA,

stage\_la\_excels.manual\_customer\_mapping\_v\_BRAZIL,

stage\_la\_excels.manual\_customer\_mapping\_v\_CHILE,

stage\_la\_excels.manual\_customer\_mapping\_v\_COLOMBIA,

stage\_la\_excels.manual\_customer\_mapping\_v\_MEXICO,

stage\_la\_excels.manual\_customer\_mapping\_v\_PERU,

stage\_la\_excels.manual\_customer\_mapping\_v\_LADMAR

* Union All combines the rows from all 7 tables into a single result set without removing any duplicates.
* Similarly, Created ‘stage\_la\_common.customer\_mapping\_analysis’ to consolidate data from the following 7 existing tables into one by using UNION ALL without removing any duplicates:

stage\_la\_common.customer\_mapping\_analysis\_ARGENTINA,

stage\_la\_common.customer\_mapping\_analysis \_BRAZIL,

stage\_la\_common.customer\_mapping\_analysis \_CHILE,

stage\_la\_common.customer\_mapping\_analysis \_COLOMBIA,

stage\_la\_common.customer\_mapping\_analysis \_MEXICO,

stage\_la\_common.customer\_mapping\_analysis \_PERU,

stage\_la\_common.customer\_mapping\_analysis \_LADMAR

### 10.5.2 02\_ARCHIVING

* Here Data is archived from the Existing tables into new Archiving tables.
* Existing tables and their corresponding Archiving tables are shown below:

|  |  |
| --- | --- |
| **EXISTING TABLE** | **ARCHIVING TABLE** |
| repo\_la\_output.sode\_cal\_time\_dim | repo\_la\_output\_archiving.sode\_cal\_time\_dim\_archiving |
| repo\_la\_output.sode\_cal\_time\_dim\_rolling | repo\_la\_output\_archiving.sode\_cal\_time\_dim\_rolling\_archiving |
| repo\_la\_output.sode\_pc064\_profit\_center\_dim | repo\_la\_output\_archiving.sode\_pc064\_profit\_center\_dim\_archiving |
| repo\_la\_output.sode\_g705\_geography\_dim | repo\_la\_output\_archiving.sode\_g705\_geography\_dim\_archiving |
| repo\_la\_output.sode\_c898\_customer\_dim | repo\_la\_output\_archiving.sode\_c898\_customer\_dim\_archiving |
| repo\_la\_output.sode\_p5005\_product\_dim | repo\_la\_output\_archiving.sode\_p5005\_product\_dim\_archiving |
| repo\_la\_output.sode\_p5801\_product\_dim\_ama | repo\_la\_output\_archiving.sode\_p5801\_product\_dim\_ama\_archiving |
| repo\_la\_output.sode\_p5801\_product\_dim | repo\_la\_output\_archiving.sode\_p5801\_product\_dim\_archiving |
| repo\_la\_output.sode\_crncy\_dim | repo\_la\_output\_archiving.sode\_crncy\_dim\_archiving |
| repo\_la\_output.manual\_customer\_mapping\_dim | repo\_la\_output\_archiving.manual\_customer\_mapping\_dim\_archiving |
| repo\_la\_output.la\_flat\_output\_fct | repo\_la\_output\_archiving.la\_flat\_output\_fct\_archiving |
| repo\_la\_output.la\_sd\_flat\_output\_fct | repo\_la\_output\_archiving.la\_sd\_flat\_output\_fct\_archiving |
| stage\_la\_excels.manual\_customer\_mapping\_v | repo\_la\_output\_archiving.manual\_customer\_mapping\_v\_archiving |
| stage\_la\_common.customer\_mapping\_analysis | repo\_la\_output\_archiving.customer\_mapping\_analysis\_archiving |

# AUTOMATED DQ CHECKS

### 11.1 GC Output - DQ - Data Duplication

**Action:** Duplicates Check: A record\_count value is taken and the count of number of individual records is checked. If the count is greater than one, then it implies that there are duplicate records present in the output.

**Expected Result:** No duplicate rows in output.

**Action on failure:** Output not shared further, and incident is raised for further investigation.

**Test case ID: 517927**

### 11.2 GC Output - DQ - NOS Completeness

**Action:** Validation if NOS columns have no 'NULL' values: All records are selected from the final output table (REPO\_LA.LA\_FLAT\_OUTPUT\_FCT) where NOS\_LC is NULL or NOS\_R$ is NULL or NOS\_C$ is NULL. In case such records exist, it implies that there are records where NOS is NULL.

**Expected Result:** There are no 'NULL' Values in 'NOS' (all currency types)

**Action on failure:** Output not shared further, and incident is raised for further investigation.

**Test case ID:** 517928

### 11.3 GC Output - DQ - Output MD Completeness

**Action:** Validation if Output Master Data have no 'NULL' values: A logic is implemented where all records are selected from the final output table (REPO\_LA.LA\_FLAT\_OUTPUT\_FCT) where geo\_id, time\_id, product\_id, customer\_id, customer\_name is NULL. In case such records exist, it implies that there are records where Master Data is NULL.

**Expected Result:** There are no 'NULL' Values in Output Master Data (Geo-Customer-Product-Time) columns (names and IDs) have values in all rows.

**Action on failure:** Output not shared further, and incident is raised for further investigation.

**Test case ID:** 517929

### 11.4 GC Output - Uploader - CMI Ingested Values

**Action:**

1. Aggregate Shipments Data on Geo - Month - Reporting Customer - Category level
2. Aggregate Uploader NSRd Manual Input and SD Manual input on Geo - Month - Category - Reporting Customer level
3. Join data from step #2 with aggregated shipments to remove input data for Geo - Month - Category - Reporting Customer that does not have any shipments in full scope of C-NOS Reporting Time frame (PFY + CFYTD)
4. Aggregate data from step #3 on Geo - Month level
5. Aggregate GC Output NSRD Manual Input and SD Manual input on Geo - Month level
6. Compare data from Step #4 with data from Step #5 on Geo - Month level

**Expected Result:**

1. Provides aggregated shipments data for reporting customers and categories at a geo and month level
2. Provides aggregated uploader NSRD manual input and SD manual input data for reporting customers and categories at a geo and month level
3. Removes inputs data for Geo - Month - Category - Reporting Customer that do not have any corresponding shipments data in the full scope of C-NOS Reporting Time frame
4. Results in inputs data with patterns only for Geo - Month - Category - Reporting Customer with corresponding shipments data
5. Provides aggregated data for Geo - Month level based on the inputs data from step #3,
6. Provides aggregated GC Output NSRD Manual Input and SD Manual input data for Geo - Month level, ensures that data from step #4 and #5 are matching at a Geo - Month level.

**Action on failure:** INC is Raised, output is not shared with business.

**Test case ID:** 517930

### 11.5 GC Output - Uploader - FMR Ingested Values

**Action:**

1. Aggregate Shipments Data on Geo - Month - Reporting Customer - Category level
2. Aggregate Uploader FMR NSRD - Base PC, FMR SD - Base PC and FMR TDC - Base PC input on Geo - Month level
3. Join data from step #2 with aggregated shipments to remove input data for Geo - Month - Category - Reporting Customer that does not have any shipments in full scope of C-NOS Reporting Time frame (PFY + CFYTD)
4. Aggregate data from step #3 on Geo - Month level
5. Aggregate GC Output FMR NSRD - Base PC (NSRD Tie Out + NSRD SAP + NSRD Manual Input + NIT), FMR SD - Base PC (SD Tie Out + SD TPR + SD Live Rates + SD Fixed Budget + SD Manual Input) and FMR TDC - Base PC input on Geo - Month level.
6. Compare data from Step #4 with data from Step #5 on Geo - Month level.

**Expected Result:**

1. Provides aggregated shipments data for reporting customers and categories at a geo and month level.
2. Provides aggregated uploader FMR NSRD - Base PC, FMR SD - Base PC and FMR TDC - Base PC input data at a geo and month level.
3. Removes inputs data for Geo - Month - Category - Reporting Customer that do not have any corresponding shipments data in the full scope of C-NOS Reporting Time frame (PFY + CFYTD).
4. Results in inputs data with patterns only for Geo - Month - Category - Reporting Customer with corresponding shipments data.
5. Provides aggregated data for Geo - Month level based on the inputs data from step #3.
6. Provides aggregated GC Output FMR NSRD - Base PC, FMR SD - Base PC and FMR TDC - Base PC input data at a geo and month level
7. Ensures that data from step #4 and #5 are matching at a Geo - Month level

**Action on failure:** Incident is Raised, output is not shared with business.

**Test case ID:** 517931

### 11.6 GC Output - Uploader - SD - TPR Ingested Values

**Action:**

1. Aggregate Uploader SD - TPR Input on SMO - Month level
2. Aggregate GC Output - TPR Input on SMO - Month level
3. Aggregate GC Output - TPR Input on SMO - Month level

**Expected Results:**

The expected result is to aggregate Uploader SD - TPR Input data and GC Output - TPR Input data at a SMO - Month level and ensure that the data matches between them.

**Action on failure:** Incident is created, output is not shared with the business.

**Test case ID:** 517932

### 11.7 GC Output - NIT Ingested Values

**Action:**

1. Sum Input NIT (OS in LC - SDED CO LE) input on Geo - Month granularity
2. Sum Output NIT on Geo - Month granularity
3. Compare Sums from Step 1 with Sum from Step 2

**Expected Result:**

Sum of NIT after aggregation to geo-monthly granularity available in input are equal vs the output.

**Action on failure:** This is a hard stop test case and will stop further data validations until such errors are resolved

**Test case ID**: 517934

### 11.8 GC Output - SD Live Rates Ingested Values

**Actions:**

1. Sum Output SD Live Rates at Geo - Month level
2. Join SD Live Rates input data with Valid\_Fund\_ID input file from Uploader to eliminate rows related for Fund\_IDs not maintained in uploader
3. Join the data from the result of step #2 with Shipments data based on the Geo-Month-Reporting Customer ID - Product granularity from lowest to highest to exclude lines of input that does not have any shipments related to them.
4. Sum outcome input data from Step #3 on Geo-Month level
5. Compare sums from Step 1 vs sum from Step 4

**Expected Results:**

1. SD Live Rates are summed and rows for Fund\_IDs not present in uploader Valid\_Fund\_ID input file are not considered further in the check.
2. Rows for the data based on the Geo-Month-Reporting Customer ID - Product granularity from (lowest to highest granularity) that do not have any shipments are not considered further in the check.
3. Data is summed and Sum of SD live rates after aggregation to geo-monthly granularity available in input are equal vs the output.

**Actions on failure**: Output not shared further, incident raised for further investigation.

**Test case ID**: 517935

### 11.9 GC Output - SD Fixed Budget Ingested Values

**Action:**

1. Sum the Output SD Fixed Budget on Geo - Month granularity
2. Join SD Fixed Budget input data with Valid\_Fund\_ID input file from Uploader to eliminate rows related for Fund\_IDs not maintained in uploader.
3. Join the SD Fixed Budget input data from step #2 with uploader Fund\_ID\_SubSectro\_Mapping input file to add product granularity input data
4. Join the SD Fixed Budget input data from Step #3 with Shipments on Geo-Reporting Customer ID - Month - SubSector level to exclude all lines where shipments does not exist in this granularity
5. Sum outcome input data from Step #4 on Geo-Month level
6. Compare sums from Step 1 vs sum from Step 5

**Expected results:**

1. SD Fixed budget is summed on Geo-Month granularity and rows for Fund\_IDs not present in uploader Valid\_Fund\_ID input file are excluded
2. Product granularity input data is added to the SD Fixed Budget input data by joining it with the uploader Fund\_ID\_SubSector\_Mapping input file
3. SD Fixed Budget input data joined with shipments data on Geo-Reporting Customer ID - Month - SubSector level, excluding all lines where shipments did not exist in this granularity
4. Data is summed and Sum of SD fixed budget after aggregation to geo-monthly granularity available in input are equal vs the output

**Action on failure:** Output is not shared further, incident raised for further investigation.

**Test case ID:** 517936

### 11.10 GC Output - GIV to GS rate – report

**Action:**

1. Aggregate the GIV on Geo - Month - Reporting Customer Level
2. Aggregate the GS on Geo - Month - Reporting Customer Level
3. Perform a delta comparison between data from Step #1 with Data from Step #2 on Geo - Month - Reporting Customer Level

**Expected results:** GIV, GS is aggregated at Geo - Month - Reporting Customer Level and on performing delta comparison the difference should be not higher than 2% for Brazil and not higher than 0.5% for other LA Geo's

**Action on failure:** Extract data in form of report for Geo-Month-Reporting Customer where delta is higher vs expected.

**Test case ID:**  545845

### 11.11 DQ\_0002\_LA\_CM

Description:

* Compare Customer L9 IDs mapped to Reporting Customer IDs from Prev Cycle to Curr Cycle and flag Reporting Customer IDs with removed / added Customer L9 IDs
* Compare list of Reporting Customer IDs from Prev Cycle to Curr Cycle and flag removed / added Reporting Customer IDs.
* Check for Duplicated Customer L9 IDs with the same Customer Description.
* Check for duplicates rows wherein a Customer L9 ID is coming in as both SAP and HUB\_SHIP.
* Compare Customer L9 IDs mapped to Reporting Customer IDs having HUB\_SHIP and SAP\_HUB\_SHIP description to the manual customer mapping generated.
* Flag if there are change in description from Previous hub file to Current hub file.

### 11.12 DQ\_0003\_LA\_Formulas

Description:

* Shows the difference calculated between NOS and GC in Output Reports, Manually Calculated NOS and GC using formula and KPIs from Outputs at Customer L9 ID - FPC ID - Month level.
* Flags the differences with greater than or less than 99%

### 11.13 DQ\_0005\_LA\_Gross\_Sales

Description:

* Shows the Gross sales difference calculated between Previous Cycle and Current Cycle at Reporting Customer ID - Subsector ID - Month level.
* Flag those with differences greater than or less than 99%

### 11.14 DQ\_0006\_LA\_NIT

Description:

* Shows the difference calculated between NIT in AAS and manually calculated NIT based on the formula at Customer L9 ID - FPC ID Level - Month Level.
* Show the difference calculated between Previous and Current Cycle at Customer L9 ID - FPC ID Level - Month Level.
* Separately Flag the differences greater than or less than 99%

### 11.15 DQ\_0007\_LA\_NSRD\_SD\_FMR

Description:

* Check Previous submissions if modified
* Check the formula for NSRd/SD Totals
* For those changed customers, raise a flag that amounts will change
* For the rest, do a high-level comparison at Subsector / Month between Prev vs current cycle
* Check Input versus Output for FMR and the Totals at Country Level (Only with Shipments for that Month).

### 11.16 DQ\_0008\_LA\_PEA

Description:

* Check timestamp on generated file at the Storage Accounts.

### 11.17 DQ\_0009\_LA\_Product\_Mapping

Description:

* Flag the Brand IDs that are excluded in Product Mapping but appear to have shipments in AAS.

### 11.18 DQ\_0010\_LA\_SD\_TPR

Description:

* Show the difference calculated between SD TPR Inputs and SD TPR from Outputs at SMO - Profit Centre - FPC ID - Month level.
* Show the difference calculated between Previous and Current Cycle at SMO - Profit Centre - Subsector ID - Month level.
* Flag those with differences greater than or less than 99% (For both checked separately).

### 11.19 DQ\_0011\_LA\_TDC\_FMR

Description:

* Show the difference calculated between Previous and Current Cycle at SMO - Profit Centre - Subsector - Month level.
* Flag those with differences greater than or less than 99%.

### 11.20 DQ\_0012\_LA\_VOL

Description:

* Show the difference calculated between Previous Cycle and Current Cycle at SMO - Profit Centre - Subsector - Month level.
* Flag those with differences greater than or less than 99%

### 11.21 DQ\_0013\_LA\_NSRD\_Manual\_Input

Description:

* Show the difference calculated between NSRD Manual Input from MMR vs NSRD Manual Input in AAS at Reporting Customer ID - Subsector ID - Month level.
* Flag those with differences greater than or less than 99%

### 11.22 DQ\_0014\_LA\_SD\_Manual\_Input

Description:

* Show the difference calculated between SD Manual Input from Uploader Template and SD Manual Input from Outputs at SMO - Reporting Customer ID - Subsector ID - Month level.
* Flag those with differences greater than or less than 99%

### 11.23 DQ\_0015\_LA\_SD\_Live\_Rates

Description:

* Flag of missing file in SharePoint folder.
* Flag that there are special characters.
* Count of how many rows that has special characters.

### 11.24 DQ\_0016\_LA\_SD\_LIVE\_RATES\_FIXED\_BUDGET

Description:

* Compare SD Live Rates/Fixed Budget of Reporting Customer ID - Subsector ID - Month combinations between Prev and Curr Cycles.
* Filtering out the Customers from CM, Fund IDs from Valid Fund ID Mapping, Fund IDs from Fund ID Subsector Mapping, and those with Shipments only, compare Inputs from Base Ship Funded Input Files to SD Live Rates/Fixed Budget values in AAS.

### 11.25 DQ\_0017\_LA\_SD\_Fixed\_Budget

Description:

* Flag of missing file in SharePoint folder
* Flag that there are special characters.
* Count of how many rows that has special characters.

### 11.26 DQ\_0018\_LA\_NIT

Description:

* Check columns os\_in\_lc and sded\_co\_le for commas (",")
* List all currencies available in NIT view.

### 11.27 DQ\_0019\_LA\_NSRD\_PROMPT\_PAYMENTS

Description:

* Show the difference calculated between Previous Cycle and Current Cycle at Reporting Customer ID - Subsector ID - Month level.
* Flag those with differences greater than or less than 99%.

### 11.28 DQ\_0020\_LA\_NSRD\_LOGISTIC\_DISCOUNTS

Description:

* Shows the difference calculated between Previous Cycle and Current Cycle at Reporting Customer ID - Subsector ID - Month level.
* Flag the entries that are having differences greater than or less than 99%.

### 11.29 DQ\_0021\_LA\_NSRD\_OTHERS

Description:

* Show the difference calculated between Previous Cycle and Current Cycle at Reporting Customer ID - Subsector ID - Month level.
* Flag those with differences greater than or less than 99%.

### 11.30 DQ\_0022\_LA\_Gross\_Sales\_Ingestion

Description:

* Timestamp of the Ingested data is retrieved to determine when data is ingested or processed.
* Count of rows in GLPCA is used to determine if there is data present. If count is more than 0, it indicates that data exists in the table.

### 11.31 DQ\_0023\_LA\_Customer\_Mapping

Description:

* Timestamp of Ingested data is retrieved to determine when data is ingested or processed.
* Flag to check whether the HUB file consists of data or not.

### 11.32 DQ\_0024\_LA\_SD\_File\_Check

Description:

* Checks if date format for column Month is DD/MM/YYYY.
* Checks if file extension format is .CSV.
* Checks if the Column Header Names are correct and in order.

### 11.33 DQ\_0025\_LA\_Regional\_GIV

Description:

* Shows the difference calculated between Previous and Current Cycle at Regional - Profit Centre - Month level.
* Flags the entries that are having differences greater than or less than 99%.

### 11.34 DQ\_0026\_LA\_Regional\_Gross\_Sales

Description:

* Shows the difference calculated between Previous and Current Cycle at Regional - Profit Centre - Month level.
* Flag the entries that are having differences greater than or less than 99%.

### 11.35 DQ\_0027\_LA\_Gross\_Sales\_Input\_Output

Description:

* Comparison is done to determine the difference calculated between input vs output at Time ID - Geo ID - Reporting Customer ID level.
* After comparing the input and output, all the entries that have differences greater than 0 are flagged.

### 11.36 DQ\_0028\_LA\_NIT\_Input\_Output

Description:

* The rows in the input that were not ingested further can be identified using the condition ‘where TimeID is not null’.
* Using the condition ‘where TimeID is not null’, the following rows are selected from the input that are not processed any further:

### 11.37 DQ\_0028\_LA\_NIT\_Input\_Output\_SP\_extract

Description:

* An Automatic extraction of NIT is performed, and the extracted data is transferred to SharePoint.

### 11.38 DQ\_0029\_LA\_SD\_TPR\_Inputs\_With\_NO\_Ships

Description:

* Processing the SD-TPR inputs with no shipments, extracts the generation data and transfers it to the SP.

### 11.39 DQ\_0030\_LA\_NIT\_Input\_Output\_Comparison

Description:

* Comparing the NIT input file with the expected output file to ensure the data is matching.
* Providing Input and output data at the Reporting customer level and Month level.

# AAS REFRESH ACTIVITY

We are having two types of AAS models:

* Regional AAS model
* Market AAS model

Regional model is refreshed by using ‘pipeline\_get\_list\_la\_refresh\_aas\_model’. This model is accessed by regional SPOC to validate the data.

Upon confirmation by SPOC for the data validation, regional model is refreshed by performing manual AAS refresh activity.

### 12.1 Steps to view AAS Models:

****

# PEA

## Common Steps for all PEA regions

Step 101 - Create NON BOM ELEMENTS

Object Created: stage\_common.pea\_non\_bom\_elements

**Source Blob/View/Table Read in Step:**

stage\_dir\_shipments.masterdata\_prod\_bom\_dim

stage\_dir\_shipments.masterdata\_prod\_life\_cycle\_dim life

stage\_dir\_shipments.masterdata\_prod\_uom\_esi uom

**Columns Read from Source or Calculated**:

* ISO\_CNTRY\_CODE
* STAGE\_ID
* PROD\_ID
* BARR\_CUR\_CODE
* UOM\_ID

**Filters:**

UOM\_ID IN ('IT','MP','BP','SW','CS')

CURR\_IND = 'Y'

BAR\_CURR\_CODE IS NOT NULL

PROD\_UOM\_END\_DATE > CURRENT\_DATE()

PROD\_LIFE\_CYCLE\_END\_DATE > CURRENT\_DATE()

CURR\_IND = 'Y’.

**Notebook used:**

**/Shared/CNGC\_COMMON/30\_PREPROCESSING\_COMMON\_DIMS/310\_CREATE\_PEA\_DIMS**

### 13.1.1 Create BOM ELEMENTS

Object Created: stage\_common.pea\_bom\_elements

Source Blob/View/Table Read in Step:

stage\_dir\_shipmentsmasterdata\_prod\_bom\_dim

joning with below tables:

stage\_dir\_shipments.masterdata\_prod\_life\_cycle\_dim life

stage\_dir\_shipments.masterdata\_prod\_uom uom

Columns Read from Source or Calculated:

* ISO\_CNTRY\_CODE
* STAGE\_ID
* PROD\_PARNT\_ID
* BARR\_CUR\_CODE
* UOM\_ID

**Filters:**

uom.UOM\_ID IN ('IT','MP','BP','SW','CS')

uom.CURR\_IND = 'Y'

uom.BAR\_CURR\_CODE IS NOT NULL

uom.PROD\_UOM\_END\_DATE > CURRENT\_DATE()

life.PROD\_LIFE\_CYCLE\_END\_DATE > CURRENT\_DATE()

life.CURR\_IND = 'Y'

life.STAGE\_6\_DATE > '2013-01-01'

bomdim.BOM\_TYPE\_CODE='M'

bomdim.PROD\_BOM\_END\_DATE > CURRENT\_DATE()

bomdim.CURR\_IND = 'Y’.

**Notebook used:**

**/Shared/CNGC\_COMMON/30\_PREPROCESSING\_COMMON\_DIMS/310\_CREATE\_PEA\_DIMS**

### 13.1.2 Create FPC to CUGTIN Mapping

Object Created: stage\_common.pea\_fpc\_to\_cugtin\_mapping

Source Blob/View/Table Read in Step:

stage\_common.pea\_non\_bom\_elements

stage\_common.pea\_bom\_elements

Columns Read from Source or Calculated:

* ISO\_CNTRY\_CODE
* STAGE\_ID
* PROD\_ID
* BARR\_CUR\_CODE
* UOM\_ID

**Notebook used:**

**/Shared/CNGC\_COMMON/30\_PREPROCESSING\_COMMON\_DIMS/310\_CREATE\_PEA\_DIMS**

### 13.1.3 FPC to GTIN Global Master List

Object Created: stage\_common.prod\_fpc\_gtin\_assoc

Create master list of all FPC to GTIN mapping.

**Sub Query which union data from following two sub results.**

1. selecting from table stage\_common.pea\_fpc\_to\_cugtin\_mapping fpc\_to\_gtin

joining with table:

stage\_dir\_shipments.masterdata\_geo\_hier\_dim geo\_hier\_dim

 2.selecting from table stage\_common.pea\_fpc\_to\_cugtin\_mapping

joining with:

 stage\_dir\_shipments.masterdata\_geo\_hier\_dim geo\_hier\_dim

 stage\_common.ship\_step\_100\_sode\_hist\_star\_work

below Columns Read from union result:

GEO\_ID

FPC\_ID

CU\_GTIN

**Filters:**

geo\_hier\_dim.GEO\_HIER\_ID = '707'

geo\_hier\_dim.CURR\_IND='Y'

geo\_hier\_dim.GEO\_HIER\_END\_DATE > CURRENT\_DATE()

fpc\_to\_gtin.STAGE\_ID NOT IN ('06','07')

**Notebook used:**

**/Shared/CNGC\_COMMON/30\_PREPROCESSING\_COMMON\_DIMS/310\_CREATE\_PEA\_DIMS**

## Steps for LA PEA Implementation

### 13.2.1 Create shipment aggregation

**Object Created: repo\_la.pea\_shipment\_agg<SMO\_NAME>**

This step creates shipment aggregation data.

**Source Blob/View/Table Read in Step:**

repo\_la.dir\_ship\_la\_fct\_lc\_<SMO\_NAME>

**Columns Read from Source or Calculated**:

FY

GEO\_5\_ID

SUB\_SECTOR

CATEGORY

BRAND

SEGMENT

BRAND\_FORM

FPC

PROD\_12\_NAME

VOLUME

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.2 Create TDC input table

**Object Created : repo\_la.pea\_mnr\_tdc\_su\_<SMO\_NAME>**

This step creates consolidated TDC input table from both Base and Export TDC files, by taking max of records from both to avoid duplicates.

**Source Blob/View/Table Read in Step:**

stage\_mmr.mnr\_tdc\_su\_dim\_vw\_v

**Columns Read from Source or Calculated:**

FY\_CODE

GEO\_ID,

FPC\_ID

PROFT\_CTR\_ID

CURRENCY\_ID

**Filters: geo\_id specific for Split SMO LA**

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.3 Joining GTIN, shipment and TDC data

**Object Created : repo\_la.pea\_data\_join\_<SMO\_NAME>**

This step creates a consolidated table from GTIN, Shipments and TDC data and below are the tables needed for this step.

**Source Blob/View/Table Read in Step:**

stage\_common.prod\_fpc\_gtin\_assoc gtin

repo\_la.pea\_shipment\_agg\_<SMO\_NAME>

repo\_la.pea\_mnr\_tdc\_su\_<SMO\_NAME>

stage\_common.SODE\_P5005\_FDIM prod\_hier\_dim

stage\_mmr.mnr\_geo\_mapng\_dim\_vw\_v

**Columns Read from Source or Calculated:**

CU\_GTIN

FPC\_ID

PROD\_4\_ID

PROD\_5\_ID

PROD\_6\_ID

PROD\_7\_ID

PROD\_8\_ID

GEO\_ID

FY\_ID

VOL

TDC\_PER\_SU

**Filters:**

PROD\_HIER\_ID = '5005'

prod\_hier\_dim.CURR\_IND = 'Y'

GEO\_HIER\_DIM.GEO\_3\_ID = '952'

GEO\_HIER\_DIM.GEO\_HIER\_ID = '707

GEO\_HIER\_DIM.CURR\_IND=’Y’

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.4 Calculating AVG TDC per GTIN

**Object Created : repo\_la.pea\_gtin\_avg\_<smo\_name>**

This step uses consolidated GTIN, SHIPMENTS and TDC table and calculates average TDC per GTIN and below are the tables needed for this step.

**Source Blob/View/Table Read in Step:**

repo\_la.pea\_data\_join\_<SMO\_NAME>

stage\_common.prod\_fpc\_gtin\_assoc

**Columns Read from Source or Calculated:**

GEO\_ID,

ISO\_CNTRY\_CODE,

FY\_ID,

SUBSC\_ID,

CATEG\_ID,

BRAND\_ID,

SEGMN\_ID,

FORM\_ID,

CU\_GTIN,

FPC\_ID,

VOL,

TDC\_SU

**Filters: Geo\_id smo specific for split la smo**

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.5 Calculating AVG TDC on GTIN level

**Object Created : repo\_la.pea\_gtin\_level\_tdc\_avg\_<SMO\_NAME>**

This step calculates avg TDC on GTIN level by using table created in step 301 and below are the tables needed for this step.

**Source Blob/View/Table Read in Step:**

repo\_la.pea\_gtin\_avg\_<SMO\_NAME>

self-joining with:

repo\_la.pea\_gtin\_avg\_<SMO\_NAME>

to limit the scope only to those rows having TDC input, otherwise averages would be underestimated.

**Columns Read from Source or Calculated:**

* GEO\_ID,
* FY\_ID,
* SUBSC\_ID,
* CATEG\_ID,
* BRAND\_ID,
* SEGMN\_ID,
* FORM\_ID,
* CU\_GTIN,
* FPC\_ID,
* VOL,
* TDC\_SU
* TDC\_SU (TDC\_SU\_CU\_GTIN)
* TDC\_SU (TDC\_SU\_BRAND\_FORM)
* TDC\_SU (TDC\_SU\_SEGMENT)
* TDC\_SU (TDC\_SU\_BRAND)
* TDC\_SU (TDC\_SU\_CATEG)

**Filters: None**

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.6 Create output on CATEGORY/BRAND/SEGMENT/BRAND FORM level

**Object Created : repo\_la.pea\_product\_levels\_tdc\_avg\_<SMO\_NAME>**

This step creates output on all PROD levels and below are the tables needed for this step.

**Source Blob/View/Table Read in Step:**

repo\_la.pea\_gtin\_level\_tdc\_avg\_<SMO\_NAME>

**Columns Read from Source or Calculated:**

* GEO\_ID,
* FY\_ID,
* PRODUCT\_ID
* PRODUCT\_LEVEL
* TDC\_SU

**Filters: None**

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.7 Create PEA LA fact table

**Object Created :** **repo\_la.pea\_la\_fct\_before\_geo\_filter\_<SMO\_NAME>**

This step creates a PEA Fact table converting TDC from USD to LC and below are the tables needed for this step.

this is an intermediate table before generating output without geo\_id filter.

**Source Blob/View/Table Read in Step:**

repo\_la.pea\_gtin\_level\_tdc\_avg\_<SMO\_NAME>

repo\_la.pea\_product\_levels\_tdc\_avg\_<SMO\_NAME>

repo\_la.pea\_tables\_refresh\_data\_< SMO\_NAME>

**Columns Read from Source or Calculated:**

* GEO\_ID,
* PRODUCT\_ID
* FPC\_ID
* CURRENCY\_ID
* FY\_ID
* TDC\_\_PER\_SU \* EXCHG\_RATE\_MULTR\_VAL
* TDC\_LAST\_REFRESH
* SH\_LAST\_REFRESH

**Filters:**

TDC\_SU IS NOT NULL AND TDC\_SU <> 0

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

### 13.2.8 Create PEA LA fact table

**Object Created :** **repo\_la.pea\_la\_fct\_before\_geo\_filter\_<SMO\_NAME>**

This step uses a temporary "before\_geo\_filter" fact table to apply a new GEO\_IDs filter

**Source Blob/View/Table Read in Step:**

repo\_la.pea\_la\_fct\_before\_geo\_filter\_<SMO\_NAME>

tech.config\_output\_geo\_ids\_4\_pea\_v con

**Columns Read from Source or Calculated:**

* GEO\_ID,
* PRODUCT\_ID
* FPC\_ID
* CURRENCY\_ID
* FY\_ID
* TDC\_\_PER\_SU \* EXCHG\_RATE\_MULTR\_VAL
* TDC\_LAST\_REFRESH
* SH\_LAST\_REFRESH

**Filters:**

REGION = 'LA'

And geo\_id is smo specific for split LA SMO

**Notebook used for allocation:**

**/Shared/CNGC\_LA\_SMO/<SMO\_NAME>/40\_PROCESSING\_REPORTING\_<SMO\_NAME>/560\_LA\_PEA\_FACT\_PROCESS\_<SMO\_NAME>**

# DD API / DAS SETUP FOR LA

* The Output generated in the ADB is shared to the downstream using DD-API, the row level security is implemented using **Data Authorization service API (DAS API).**

## Creation / Registration of Logical dataset

* For each physical dataset in the databricks which is to be published, a logical dataset is created in the DD-API.

This is created by calling a function the following function.

create\_logical\_object('Customer NOS & Gross Contribution Light Refined','CNOS\_GC\_EUROPE\_CNOS\_Report','CNOSGC','CDL' )

This function is a part of LLMC (Low Level metadata client)

cdl\_common\_utils.metadata\_client.metadata\_client import LowLevelMetadataClient.

* To publish a new dataset to DD-API, it must be registered using the below method.

create\_logical\_object(<owning\_application\_name>, <object\_type>, <logical\_object\_name>, <data\_type\_code>, <data\_provider\_code>)

owning\_application\_name à Registered name of the application with DD-API

object\_type

object\_name à No bound-on naming convention (can be decided with business needs)

data\_provider\_code à Irrespective of the Application this parameter is always ‘CDL’.

data\_type\_code à This varies application to application (must be obtained from DD-API team)

* For Latin America we have the following objects which are published:

|  |  |
| --- | --- |
| **logicalTables** | **in\_path** |
| la\_sd\_flat\_output\_fct | repo\_la\_output.la\_sd\_flat\_output\_fct |
| la\_flat\_output\_fct | repo\_la\_output.la\_flat\_output\_fct |
| L9\_PHC\_OUTPUT | stage\_europe.L9\_PHC\_OUTPUT |

Where in path à ADB Object Name LogicalTables à Reference dataset name in the DD-API

## DAS API Setup

* DAS API is used to get the row level configuration for each subset of data present in the dataset.
* For each subset of data, there is a permission ID maintained in the metadata. When DAS API is invoked using this permission ID, it returns a secure group Key (Permission Key).
* In DAS API backend, Azure Active Directory Groups are tagged to one or multiple permission Keys, only those Groups can access data associated to that permission key.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **permission\_id** | **permission\_string** | **logicalTables** | **in\_path** | **natural\_key\_val** | **out\_path** |
| CDL-CNGC-DDAPI-LA\_SMO\_Restricted\_032 | GEO\_ID=032 | la\_sd\_flat\_output\_fct | repo\_la\_output.la\_sd\_flat\_output\_fct | GEO\_ID | /mnt/cngc-outgoing/light-refined/LA/CNOS\_GC\_LA\_SD\_Restricted\_Report |
| CDL-CNGC-DDAPI-LA\_SMO\_Restricted\_032 | GEO\_ID=032 | la\_flat\_output\_fct | repo\_la\_output.la\_flat\_output\_fct | GEO\_ID | /mnt/cngc-outgoing/light-refined/LA/CNOS\_GC\_LA\_Restricted\_Report |
| CDL-CNGC-DDAPI-EU-PHC\_PHC\_NOS | RLS=PHC\_NOS | L9\_PHC\_OUTPUT | stage\_europe.L9\_PHC\_OUTPUT | RLS | /mnt/cngc-outgoing/light-refined/Europe/CNOS\_EU\_PHC |

* In the table Above, we can see that for permission ID **‘CDL-CNGC-DDAPI-LA\_SMO\_Restricted\_032’,** we have a permission string GEO\_ID=032, in the publishing process, the Permission Key obtained via DAS API will be joined with all the rows that have GEO\_ID=032.
* A column which is used to join data with the permission key is known as natural\_key\_val, this is the column on basis of which data will be partitioned in the DD-API.
* A new column ‘**secure\_group\_key**’ will be created to hold the value of the permission key.
* In the later part of the publishing code, data will be partitioned using the secure Group Key and saved in the location “out\_path/secure\_group\_key=xxx”. In our case it will be ‘**/mnt/cngc-outgoing/light-refined/LA/CNOS\_GC\_LA\_SD\_Restricted\_Report/secure\_group\_key=xxxxxx’**
* Further the DD-API is invoked, via LLMC (Low Level Metadata Client) using method

create\_physical\_partitions.

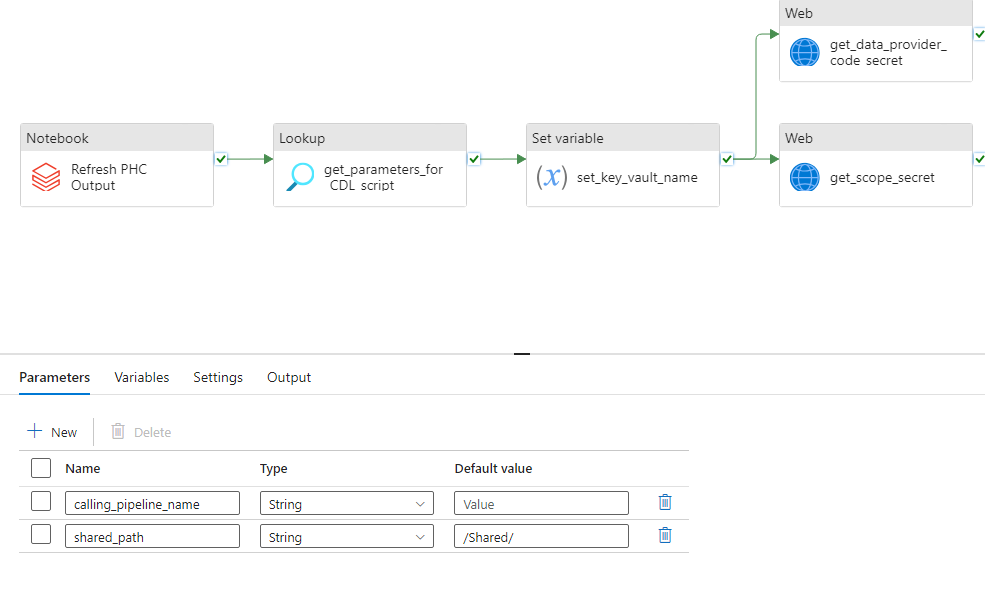
This method will notify the DD-API to register the newly created partitions and start a file mover action to deliver the dataset to relevant containers.

* LLMC also returns a process run key, this can be optionally used for partitioning, but for the most part it is used for tracking the delivery in Turbine.

## Pipeline and Notebook Configuration in ADF

**Pipeline Name:** 56\_CDL\_light\_refined/pipeline\_export\_data\_to\_CDL

**Notebook Path:** /Shared/Common\_postprocessing/CDL Publishing/CDL Publishing



The Pipeline is triggered from a parent pipeline which refreshes the LA Notebooks, two parameters are passed,

calling\_pipeline\_name à which filters metadata based on market (example if calling pipeline is attributed to LA region, lookup will fetch only the LA related datasets.)

Shared\_path à Partial path of ADB Notebook.

Activities:

Refresh PHC Output à PHC is one of the datasets which is shared via DD-API, this is a call to the PHC notebook in ADB to refresh its contents.

Lookup (get\_parameter\_for CDL script) à This lookup will fetch the names of the ADB objects to be published.

CDL Publishing Script:

The CDL Publishing Script (/Shared/Common\_postprocessing/CDL Publishing/CDL Publishing) is called with following parameters.

A screenshot of a computer

Description automatically generated

In\_path à ADB Object Path

Out\_path à Save path (This is the source of DD-API File Mover)

Natural\_key à Set of columns on basis of which, script will partition the Data and apply RLS.

Data\_provider\_code à CDL constant

Scope à Name of the key-vault scope to access secrets.

Metadata\_location à The DAS API requires input parameters (permission ID’s) to fetch the permission keys, all this is stored in SQL server, and it is copied in blob container as parquet using the following activities.

A screenshot of a computer

Description automatically generated

The parquet file is then ingested into ADB and processed.

## Publishing script Configurations in ADB

### 14.4.1 CMD 1 - Installation of required Library

Installation of ADAL which is required for Authentication (OAuth 2.0)

### 14.4.2 CMD 2 - Creation of required Widgets

* dbutils.widgets.removeAll(): This line removes all existing widgets. Widgets in Databricks are interactive elements that allow users to input values or make selections in notebooks.
* dbutils.widgets.text("scope","scope-cngc-kv-bfxd"): This line creates a text widget named "scope" with the default value set to "scope-cngc-kv-bfxd". Users can input a different value for this widget when running the notebook.
* dbutils.widgets.text("data\_provider\_code","CNOSGC"): This line creates another text widget named "data\_provider\_code" with the default value set to "CNOSGC".
* dbutils.widgets.text("in\_path","stage\_europe.eur\_rep\_output\_vw"): This line creates a text widget named "in\_path" with the default value set to "stage\_europe.eur\_rep\_output\_vw". It specifies the path from which the data will be read. This path represents a table.
* dbutils.widgets.text("out\_path","cngc-outgoing/light-refined/eur\_fpc\_lvl\_pea\_output"): This line creates a text widget named "out\_path" with the default value set to "cngc-outgoing/light-refined/eur\_fpc\_lvl\_pea\_output". It specifies the path where the processed data will be saved. The path should start with "/mnt", and if it doesn't, the code will add it during processing.
* dbutils.widgets.text("natural\_key","SMO,Geo\_ID"): This line creates a text widget named "natural\_key" with the default value set to "SMO,Geo\_ID". It represents a list of column names separated by commas and is used to assign the secure\_group\_key.
* dbutils.widgets.text("metadat\_location","/mnt/cngc-storage/processing/cdl\_publishing\_config/meta\_data\_ama\_la"): This line creates a text widget named "metadata\_location" with the default value set to "/mnt/cngc-storage/processing/cdl\_publishing\_config/meta\_data\_ama\_la". It specifies the location where metadata related to the processing will be stored.

These widgets allow users to customize these parameters when executing the notebook, providing flexibility and interactivity in the data processing workflow.

### 14.4.3 CMD 3 – Importing required libraries, Secrets, and creation of LowLevelMetadataClient

#### Importing Libraries

The requests library is imported for making HTTP requests.

Various modules and classes from custom libraries (cdl\_common\_utils.metadata\_client.metadata\_process\_run\_client, cdl\_common\_utils.metadata\_client.metadata\_client, and cdl\_common\_utils.metadata\_client.metadata\_constants) are imported.

#### Getting Widget Values

The scope variable is set using the value retrieved from a Databricks widget named "scope."

#### Getting Secrets

* Several secrets are retrieved using the specified scope:
* tenant\_id is obtained from the secret with the key "tenant-id."
* application\_id is obtained from the secret with the key "application-id."
* application\_secret is obtained from the secret with the key "sp-cnosgc-ddapi."
* metadata\_api\_application\_id is obtained from the secret with the key "metadata-api-application-id."
* metadata\_api\_url is obtained from the secret with the key "metadata-api-url."
* infrastructure\_id is obtained from the secret with the key "infrastructure-id."

#### Creating Metadata Client

* An instance of LowLevelMetadataClient is created with the retrieved secrets and connection parameters, allowing interaction with the metadata API.
* Creating Process Run Metadata:
* An instance of ProcessRunMetadata is created using the previously created LowLevelMetadataClient. This instance is used to work with the metadata API in a user-friendly way.

### 14.4.4 CMD 4 – Authentication (get\_access\_token)

#### Imported Libraries

* The function imports necessary libraries: requests, json, and adal.

#### Getting Widget Values and Secrets

* The function obtains the scope value from a Databricks widget named "scope."
* tenant\_id, client\_id, client\_secret, and resource values are retrieved from Databricks secrets using the specified scope.

#### Constructing the Token Request

* The function constructs the URL for token retrieval using the obtained tenant\_id.
* The payload includes grant type, client ID, client secret, and resource to be sent as part of the request.

#### Making a POST Request

* The function sends a POST request to the constructed URL with the specified headers and payload.
* The response contains the access token, which is extracted from the JSON response and returned by the function.
* This function essentially acts as a helper to obtain an access token necessary for authenticating with a specified resource, allowing further authenticated API requests to be made.

### 14.4.5 CMD 5 - create\_das\_payload

#### Imported Libraries

* The function uses the SparkSession and several functions from the pyspark.sql module: concat\_ws, collect\_list, concat, and lit.
* The json library is used for working with JSON data.

#### Loading Metadata

* The function reads metadata from a specified location (retrieved from a Databricks widget named "metadat\_location"). The metadata is assumed to be in Parquet format.
* It filters the DataFrame to get rows where the "in\_path" column matches the provided in\_path.

#### Aggregating Data

* The function groups the filtered DataFrame by the "logicalTables" column.
* It constructs a JSON object for each group, combining various literals and DataFrame columns to create a valid JSON structure. The resulting JSON represents an authorization object for the DAS API.

#### Extracting Payload

* The function extracts the constructed JSON payload from the DataFrame and converts it to a Python dictionary using json.loads().
* The final JSON payload is returned as a dictionary.

This function essentially processes metadata from the specified location, aggregates it, and formats it into a JSON payload suitable for use with the DAS API. The payload contains information about logical tables, application name, logical table names, and corresponding permission IDs.

### 14.4.6 CMD 6 – Importing Essential Libraries

import requests.

from requests.exceptions import HTTPError

from pyspark.sql import functions as f

import json.

from cdl\_common\_utils.common import logger

* By importing these libraries and modules, you're setting up your script to make HTTP requests, handle exceptions, work with JSON data, utilize Spark SQL functions, and log information using a custom logging module.

### 14.4.7 CMD 7 - distinct\_natural\_key

#### Selecting Columns

* The function selects the columns specified in the natural\_key list from the input DataFrame data.

#### Distinct Rows

* It then applies the distinct() function to the selected columns, removing duplicate rows based on the specified natural key columns.

#### Ordering Rows

* The orderBy() function is used to sort the distinct rows based on the columns specified in the natural\_key list.

#### Return Statement

* The sorted and distinct DataFrame is returned by the function.

In summary, this function ensures that the input DataFrame contains only distinct rows based on the specified natural key columns and returns the sorted result.

### 14.4.8 CMD 8 - get\_das\_api\_url

The get\_das\_api\_url function takes a scope parameter as input and retrieves the DAS (Data Authorization Service) API URL from the specified scope in Azure Key Vault.

#### Input Parameter

* The function takes a single parameter scope, which represents the scope in Azure Key Vault from which the DAS API URL is to be retrieved.

#### Secret Retrieval

* Using dbutils.secrets.get(scope=scope, key="das-code-api-url"), the function retrieves the DAS API URL from the specified scope in Azure Key Vault using the key "das-code-api-url".

#### Return Statement

* The retrieved API URL is returned by the function.

In summary, this function acts as a utility to obtain the DAS API URL from Azure Key Vault based on the provided scope.

### 14.4.9 CMD 9 - parse\_das\_response

* The parse\_das\_response function takes a JSON response (body) as input and converts it into a DataFrame.

#### Input Parameter

* The function takes a single parameter body, which represents the JSON response received from an API call.

#### JSON to DataFrame Conversion

* sc.parallelize(body) is used to create an RDD (Resilient Distributed Dataset) from the input JSON data.
* spark.read.json(...) reads the JSON data from the RDD and converts it into a DataFrame.

#### Return Statement

* The resulting DataFrame containing the parsed JSON data is returned by the function.

In summary, this function provides a way to convert a JSON response into a Spark DataFrame, making it easier to work with the response data in a tabular format within the Spark environment.

### 14.4.10 CMD 10 - request\_das\_api

* The request\_das\_api function makes a GET request to the DAS (Data Authorization Service) API using a provided payload.

#### Input Parameter

* The function takes a payload parameter, which represents the payload to be sent in the API request. The payload is expected to be in JSON format.

#### Logging

* The function logs information about the API request, including the payload, using the logger.info() function.

#### GET Request to DAS API

* The function constructs the request headers, including the "Content-Type" and the authorization token obtained from the get\_access\_token() function.
* It sends a GET request to the DAS API URL (obtained from get\_das\_api\_url(scope)) using the requests.request() function.
* The API response is encoded as UTF-8.

#### Response Processing

* If the request is successful, the response JSON is parsed using json.loads().
* The "permissions" field from the JSON response is extracted and passed to the parse\_das\_response() function to convert it into a DataFrame.
* The resulting DataFrame containing the parsed response data is returned.

In summary, this function sends a GET request to the DAS API, processes the response, and returns the extracted "permissions" data as a DataFrame. It also handles potential errors that might occur during the API request.

### 14.4.11 CMD 11 - check\_columns\_exists\_in\_data

* The check\_columns\_exists\_in\_data function checks if the columns specified in the natural\_key list exist in the provided DataFrame data.

#### Input Parameters

* The function takes two parameters: data, which represents the DataFrame to be checked, and natural\_key, which is a list of column names to be verified in the DataFrame.

#### Column Existence Check

* The function iterates through each column name (nk) in the natural\_key list.
* For each column, it checks if the uppercase version of the column name exists in the uppercase version of the schema of the data DataFrame. If any column from natural\_key is not found in the DataFrame's schema, the function returns False.

#### Return Statement

* If all columns specified in the natural\_key list exist in the DataFrame's schema, the function returns True.
* If the natural\_key list is empty, the function also returns True.
* In summary, this function verifies the existence of specified columns in the input DataFrame and returns True if all columns exist or if the natural\_key list is empty. Otherwise, it returns False.

### 14.4.12 CMD 12 - lookup\_das

The lookup\_das function performs the following tasks:

#### Checking Column Existence

* It checks if the columns specified in natural\_key exist in the input data DataFrame using the check\_columns\_exists\_in\_data function.

#### API Request and Data Merging

* If the columns exist, the function creates a DAS API payload using the create\_das\_payload function and makes a request to the DAS API using the request\_das\_api function.
* The response from the API is merged with metadata (df\_perm\_sting\_metadata) based on the "permission\_id" column.
* The merged DataFrame is processed to extract key-value pairs from the "permission\_string" column, and a mapping is created between "secure\_group\_key" and other columns.

#### Joining Data

* The input data DataFrame is joined with the mapping DataFrame using the columns specified in natural\_key.
* If natural\_key is empty, a default value of 0 is assigned to the "secure\_group\_key" column.

#### Return Values

* The function returns a tuple containing a return code (either "SUCCESS" or "FAIL") and the output DataFrame (output\_rls), which represents the merged data with the DAS API response.

This function handles various scenarios, such as column existence checks, API requests, data merging, and error handling, providing a comprehensive workflow for integrating data from the DAS API with the input data.

### 14.4.13 CMD 13 – Partitioning data and saving the data in storage containers.

#### Input Parameter Processing

* The script begins by retrieving various input parameters using dbutils.widgets.get(). These parameters include scope, data\_provider\_code, in\_path, out\_path, and natural\_key.

#### Process Run Key Retrieval

* The get\_process\_run\_key() function is called to obtain a unique process run key (process\_run\_key) from the metadata API. This key is used to track the execution of the entire ETL process.

#### Data Loading and Validation

* The read\_file() function loads data from the specified in\_path.
* The dq\_check\_data\_existence() function checks if the loaded data exists and is non-empty.

#### Data Processing and Transformation

* The script checks the existence of the out\_path and modifies it if necessary.
* The lookup\_das() function is called to assign secure group keys to the data. This function interacts with a DAS API to retrieve secure group keys based on natural keys.
* The script then attempts to register logical and physical metadata objects based on the processed data.

#### Partitioning and Saving Data

* The get\_partition\_list() function is used to determine partitioning based on the specified partition\_by columns.
* The save\_file() function saves the processed data to the specified out\_path partitioned by the specified columns.

#### Metadata Management

* Metadata operations include creating logical and physical objects, obtaining physical object keys, and registering physical partitions.

#### Process Run Key Status Update

* Depending on the success or failure of the ETL process, the status of the process run key is updated using change\_prk\_to\_finished().

#### Logging and Exception Handling

* The script utilizes logging extensively to capture detailed information about the execution flow.
* Exception handling is implemented to handle errors and log them for further analysis.

#### Main Execution Logic

* The main() function orchestrates the entire ETL process. It ensures that each step is executed in the correct order, handling exceptions, and logging relevant information at each stage.

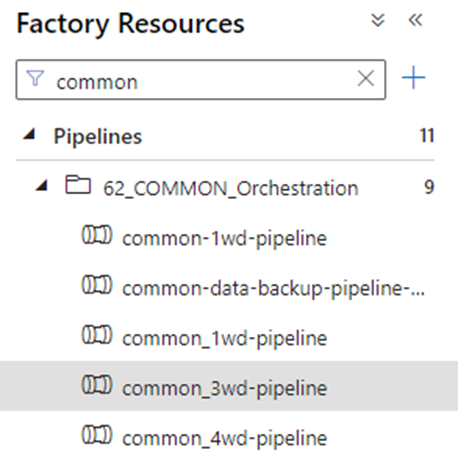
#### Return Values

* The script returns a tuple containing the processing status code (return\_code) and the process run key (process\_run\_key) for monitoring and debugging purposes.

# MASTER DATA FOR UPLOADER

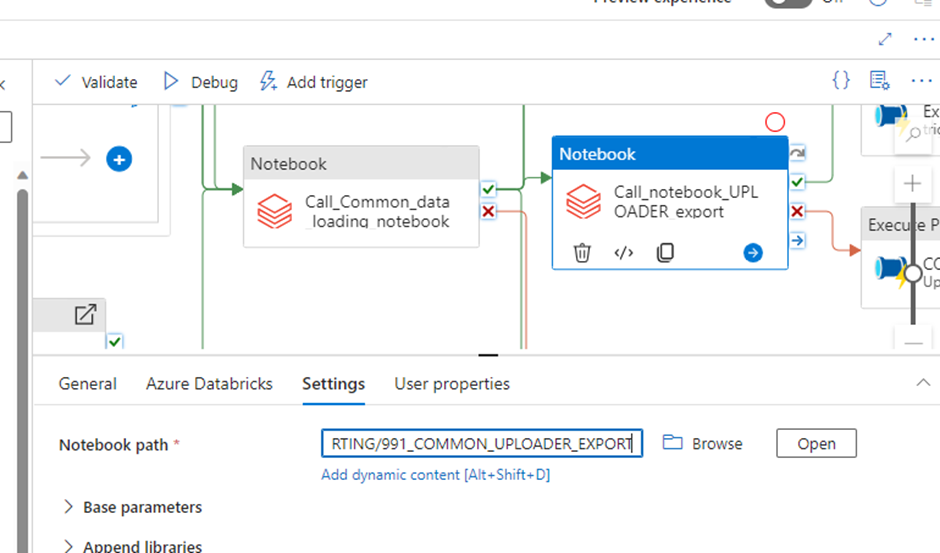
The transfer of master data from CNGC layer to uploader environment is carried out by executing pipelines in common layer. The relevant logic is explained as below:

Pipeline location in ADF:



Located in 62\_common\_Orchestration folder and in common\_3wd\_pipeline.

Inner activity:



The export of data is handled by following notebooks:

/Shared/CNGC\_COMMON/99\_REPORTING/991\_COMMON\_UPLOADER\_EXPORT

/Shared/CNGC\_COMMON/10\_INGESTION/MDM\_PREPROCESSING/300\_OUTCOME\_DATA\_EXPORT

**Step1: 300\_OUTCOME\_DATA\_EXPORT**

This is a pre-requisite step before exporting data to the uploader. Legacy code is built in following way:

Python function is built to create folder in following path in cngc-storage

cngc-storage/MDM\_outgoing\_extracts. Where the Hierarchies data needs to be copied.

In the next step,

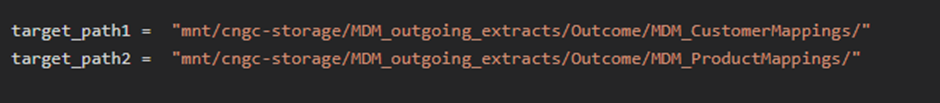
Customer mapping/Product mappings are extracted from tables

stage\_product\_mapping\_exc\_jpn\_mdm.prod\_map\_geo\_mapping\_final

stage\_la\_mdm.la\_customer\_mapping\_final

and stored into query result.

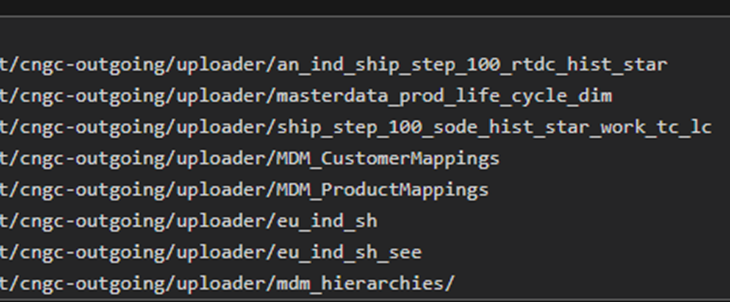
Then the files were copied to following target path into blob location:



Addition to that the source tables used above is created in notebook 350\_CREATE\_COMMON\_SHIP\_FACTS\_REF. which is a common layer code. Which sources data from shipments ingested from refined layer.

**Step 2:** **991\_COMMON\_UPLOADER\_EXPORT**

First the old files were removed from following location:



In next step

stage\_common.ship\_step\_100\_sode\_hist\_star\_work\_tc\_lc\_export.

object is created for exporting the shipment data

source used:

stage\_common.ship\_step\_100\_sode\_hist\_star\_work\_tc\_lc

then notebook 205\_MDM\_UPLOADER\_EXPORT\_VIEWS is called to export the hierarchies, such as, Prod Hierarchies, Cust hierarchies, Profit center hierarchies and geo hierarchies. After this the complete set of data is exported to the uploader layer for which the uploader tool can access the files. And upon running the load process the CNGC layer data is made available in the uploader layer.



# DATABASE OBJECT DAIGRAMS:



# DATABASE OBJECTS DEFINATIONS:





# LIST OF OBJECTS DEFNATIONS IN AAS in AAS



# AAS MODELS(Available, Configuration in AAS model, KPIs creation on fly in AAS)

## List of AAS Models in used in LA:

* LA\_CNGC\_Regional
* LA\_CNGC\_Market
* LA\_CNGC\_Market\_Archive

## Configuration of AAS:

* Location: EAST US
* Pricing tier: S4

## MODEL CONFIGURATIONS:

### LA\_CNGC\_Regional

* Data Source: SQL/cngc-sqlserver database windows net;cngc-sqldb
* Refresh Mode: Clear 🡪 Process Full
* Refreshed via: Orchestrated via ADF

#### List of Datasets used

|  |  |
| --- | --- |
| **Table Name** | **Hidden** |
| CNGC\_LA\_OUTPUT\_RSMO\_FACT | No |
| CNGC\_LA\_SD\_FACT | No |
| CORP\_CUSTOMER\_DIM | No |
| currency\_dim | Yes |
| GEOGRAPHY\_DIM | No |
| MANNUAL\_CUSTOMER\_MAPPING | No |
| PRODUCT\_DIM | No |
| PROFIT\_CENTER\_DIM | No |
| rls\_security | Yes |
| TIME\_DIM | No |
| time\_dim\_rolling | Yes |
|  |  |

#### Details of columns for each dataset.

##### CNGC\_LA\_OUTPUT\_RSMO\_FACT

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | FALSE | TIME\_ID |
| CUSTOMER\_ID | FALSE | CUSTOMER\_ID |
| GEO\_ID | FALSE | GEO\_ID |
| PRODUCT\_ID | FALSE | PRODUCT\_ID |
| CURRENCY\_ID | FALSE | CURRENCY\_ID |
| PROFIT\_CENTER\_ID | FALSE | PROFIT\_CENTER\_ID |
| NIV\_AMT | FALSE | NIV\_AMT |
| NIV2\_AMT | FALSE | NIV2\_AMT |
| REPORTING\_CUSTOMER\_ID | FALSE | REPORTING\_CUSTOMER\_ID |
| REPORTING\_CUSTOMER | FALSE | REPORTING\_CUSTOMER\_NAME |
| REGIONAL\_CHANNEL | FALSE | REGIONAL\_CHANNEL |
| CUST\_KEY | FALSE | CUST\_KEY |
| PROD\_KEY | FALSE | PROD\_KEY |
| RLS\_KEY | FALSE | RLS\_KEY |
| NIT\_AMT\_LC | FALSE | NIT\_AMT\_LC |
| NIT\_AMT\_RUSD | FALSE | NIT\_AMT\_RUSD |
| NIT\_AMT\_CUSD | FALSE | NIT\_AMT\_CUSD |
| TDC\_AMT\_LC | FALSE | TDC\_AMT\_LC |
| TDC\_AMT\_RUSD | FALSE | TDC\_AMT\_RUSD |
| TDC\_AMT\_CUSD | FALSE | TDC\_AMT\_CUSD |
| NSRD\_DAMAGE\_RETURNS\_AMT\_LC | FALSE | NSRD\_DAMAGE\_RETURNS\_AMT\_LC |
| NSRD\_DAMAGE\_RETURNS\_AMT\_RUSD | FALSE | NSRD\_DAMAGE\_RETURNS\_AMT\_RUSD |
| NSRD\_DAMAGE\_RETURNS\_AMT\_CUSD | FALSE | NSRD\_DAMAGE\_RETURNS\_AMT\_CUSD |
| NSRD\_PRICE\_ADJUSTMENTS\_AMT\_LC | FALSE | NSRD\_PRICE\_ADJUSTMENTS\_AMT\_LC |
| NSRD\_PRICE\_ADJUSTMENTS\_AMT\_RUSD | FALSE | NSRD\_PRICE\_ADJUSTMENTS\_AMT\_RUSD |
| NSRD\_PRICE\_ADJUSTMENTS\_AMT\_CUSD | FALSE | NSRD\_PRICE\_ADJUSTMENTS\_AMT\_CUSD |
| NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_LC | FALSE | NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_LC |
| NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_RUSD | FALSE | NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_RUSD |
| NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_CUSD | FALSE | NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_CUSD |
| NSRD\_OTHERS\_AMT\_LC | FALSE | NSRD\_OTHERS\_AMT\_LC |
| NSRD\_OTHERS\_AMT\_RUSD | FALSE | NSRD\_OTHERS\_AMT\_RUSD |
| NSRD\_OTHERS\_AMT\_CUSD | FALSE | NSRD\_OTHERS\_AMT\_CUSD |
| NSRD\_PROMPT\_PAYMENT\_AMT\_LC | FALSE | NSRD\_PROMPT\_PAYMENT\_AMT\_LC |
| NSRD\_PROMPT\_PAYMENT\_AMT\_RUSD | FALSE | NSRD\_PROMPT\_PAYMENT\_AMT\_RUSD |
| NSRD\_PROMPT\_PAYMENT\_AMT\_CUSD | FALSE | NSRD\_PROMPT\_PAYMENT\_AMT\_CUSD |
| NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_LC | FALSE | NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_LC |
| NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_RUSD | FALSE | NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_RUSD |
| NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_CUSD | FALSE | NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_CUSD |
| NSRD\_WRITE\_OFF\_AMT\_LC | FALSE | NSRD\_WRITE\_OFF\_AMT\_LC |
| NSRD\_WRITE\_OFF\_AMT\_RUSD | FALSE | NSRD\_WRITE\_OFF\_AMT\_RUSD |
| NSRD\_WRITE\_OFF\_AMT\_CUSD | FALSE | NSRD\_WRITE\_OFF\_AMT\_CUSD |
| SAP\_GROSS\_SALES\_AMT\_LC | FALSE | SAP\_GROSS\_SALES\_AMT\_LC |
| SAP\_GROSS\_SALES\_AMT\_RUSD | FALSE | SAP\_GROSS\_SALES\_AMT\_RUSD |
| SAP\_GROSS\_SALES\_AMT\_CUSD | FALSE | SAP\_GROSS\_SALES\_AMT\_CUSD |
| CAT\_NSRD\_TIE\_OUT\_LC | FALSE | CAT\_NSRD\_TIE\_OUT\_LC |
| CAT\_SD\_TIE\_OUT\_LC | FALSE | CAT\_SD\_TIE\_OUT\_LC |
| CAT\_TDC\_TIE\_OUT\_LC | FALSE | CAT\_TDC\_TIE\_OUT\_LC |
| TIE\_OUT\_NSRD\_LC | FALSE | TIE\_OUT\_NSRD\_LC |
| TIE\_OUT\_NSRD\_RUSD | FALSE | TIE\_OUT\_NSRD\_RUSD |
| TIE\_OUT\_NSRD\_CUSD | FALSE | TIE\_OUT\_NSRD\_CUSD |
| TIE\_OUT\_SD\_LC | FALSE | TIE\_OUT\_SD\_LC |
| TIE\_OUT\_SD\_RUSD | FALSE | TIE\_OUT\_SD\_RUSD |
| TIE\_OUT\_SD\_CUSD | FALSE | TIE\_OUT\_SD\_CUSD |
| TIE\_OUT\_TDC\_LC | FALSE | TIE\_OUT\_TDC\_LC |
| TIE\_OUT\_TDC\_RUSD | FALSE | TIE\_OUT\_TDC\_RUSD |
| TIE\_OUT\_TDC\_CUSD | FALSE | TIE\_OUT\_TDC\_CUSD |
| NOS\_C\_LC | FALSE | NOS\_LC |
| NOS\_C\_RUSD | FALSE | NOS\_RUSD |
| NOS\_C\_CUSD | FALSE | NOS\_CUSD |
| GC\_C\_LC | FALSE | GC\_LC |
| GC\_C\_RUSD | FALSE | GC\_RUSD |
| GC\_C\_CUSD | FALSE | GC\_CUSD |
| NSRD\_TOTAL\_C\_LC | FALSE | NSRD\_TOTAL\_LC |
| NSRD\_TOTAL\_C\_CUSD | FALSE | NSRD\_TOTAL\_CUSD |
| NSRD\_TOTAL\_C\_RUSD | FALSE | NSRD\_TOTAL\_RUSD |
| SD\_TOTAL\_C\_LC | FALSE | SD\_TOTAL\_LC |
| SD\_TOTAL\_C\_CUSD | FALSE | SD\_TOTAL\_CUSD |
| SD\_TOTAL\_C\_RUSD | FALSE | SD\_TOTAL\_RUSD |
| SMO | FALSE | SMO |
| GIV\_C\_LC | FALSE | GIV\_LC |
| GIV\_C\_RUSD | FALSE | GIV\_RUSD |
| GIV\_C\_CUSD | FALSE | GIV\_CUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_LC | FALSE | NSRD\_MANUAL\_INPUT\_AMT\_LC |
| NSRD\_MANUAL\_INPUT\_AMT\_RUSD | FALSE | NSRD\_MANUAL\_INPUT\_AMT\_RUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_CUSD | FALSE | NSRD\_MANUAL\_INPUT\_AMT\_CUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_YA\_LC | FALSE | NSRD\_MANUAL\_INPUT\_AMT\_YA\_LC |
| NSRD\_MANUAL\_INPUT\_AMT\_YA\_RUSD | FALSE | NSRD\_MANUAL\_INPUT\_AMT\_YA\_RUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_YA\_CUSD | FALSE | NSRD\_MANUAL\_INPUT\_AMT\_YA\_CUSD |
| VOLUME\_C | FALSE | VOLUME |
| CUSTOMER\_MAPPING\_KEY | FALSE | CUSTOMER\_MAPPING\_KEY |
| ATTR29 | TRUE | ATTR29 |
| ATTR17 | TRUE | ATTR17 |
| ATTR19 | TRUE | ATTR19 |
| ATTR18 | TRUE | ATTR18 |
| ATTR2 | TRUE | ATTR2 |
| ATTR3 | TRUE | ATTR3 |
| ATTR1 | TRUE | ATTR1 |
| REGIONAL\_SUB\_CHANNEL | FALSE | REGIONAL\_SUB\_CHANNEL |
| REGIONAL\_MACRO\_CHANNEL | FALSE | REGIONAL\_MACRO\_CHANNEL |
| REGIONAL\_CUSTOMER\_NAME | FALSE | REGIONAL\_CUSTOMER\_NAME |
| SAP\_CUSTOMER\_GROUP\_ID | FALSE | SAP\_CUSTOMER\_GROUP\_ID |
| CUSTOMER\_CHANNEL | FALSE | CUSTOMER\_CHANNEL |
| CUSTOMER\_SUB\_CHANNEL | FALSE | CUSTOMER\_SUB\_CHANNEL |
| CUSTOMER\_NAME | FALSE | CUSTOMER\_NAME |
| ORG\_AREA\_NAME | FALSE | ORG\_AREA\_NAME |

##### CNGC LA\_SD\_FACT

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | TRUE | TIME\_ID |
| CUSTOMER\_ID | TRUE | CUSTOMER\_ID |
| GEO\_ID | TRUE | GEO\_ID |
| PRODUCT\_ID | TRUE | PRODUCT\_ID |
| CURRENCY\_ID | TRUE | CURRENCY\_ID |
| RLS\_KEY | TRUE | RLS\_KEY |
| SD\_TPR\_AMT\_LC | TRUE | SD\_TPR\_AMT\_LC |
| SD\_TPR\_C\_RUSD | TRUE | SD\_TPR\_RUSD |
| SD\_TPR\_C\_CUSD | TRUE | SD\_TPR\_CUSD |
| SD\_LIVE\_RATES\_AMT\_LC | TRUE | SD\_LIVE\_RATES\_AMT\_LC |
| SD\_LIVE\_RATES\_C\_RUSD | TRUE | SD\_LIVE\_RATES\_RUSD |
| SD\_LIVE\_RATES\_C\_CUSD | TRUE | SD\_LIVE\_RATES\_CUSD |
| SD\_FIXED\_BUDGET\_AMT\_LC | TRUE | SD\_FIXED\_BUDGET\_AMT\_LC |
| SD\_FIXED\_BUDGET\_C\_RUSD | TRUE | SD\_FIXED\_BUDGET\_RUSD |
| SD\_FIXED\_BUDGET\_C\_CUSD | TRUE | SD\_FIXED\_BUDGET\_CUSD |
| SD\_MANUAL\_INPUT\_C\_LC | TRUE | SD\_MANUAL\_INPUT\_LC |
| SD\_MANUAL\_INPUT\_C\_RUSD | TRUE | SD\_MANUAL\_INPUT\_RUSD |
| SD\_MANUAL\_INPUT\_C\_CUSD | TRUE | SD\_MANUAL\_INPUT\_CUSD |
| CUSTOMER\_MAPPING\_KEY | TRUE | CUSTOMER\_MAPPING\_KEY |
| PROFIT\_CENTER\_ID | TRUE | PROFIT\_CENTER\_ID |

##### PRODUCT\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| PRODUCT\_ID | TRUE | PRODUCT\_ID |
| FPC | FALSE | FPC\_NAME |
| BRAND\_FORM\_ID | FALSE | BRAND\_FORM\_ID |
| BRAND\_ID | FALSE | BRAND\_ID |
| CATEGORY\_ID | FALSE | CATEGORY\_ID |
| SUBSECTOR\_ID | FALSE | SUB\_SECTOR\_ID |
| BRAND\_FORM | FALSE | BRAND\_FORM\_NAME |
| BRAND | FALSE | BRAND\_NAME |
| CATEGORY | FALSE | CATEGORY\_NAME |
| FPC\_ID | FALSE | FPC\_ID |
| SUBSECTOR | FALSE | SUB\_SECTOR\_NAME |

##### TIME\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | TRUE | TIME\_ID |
| MONTH | FALSE | MONTH |
| QUARTER | FALSE | QUARTER |
| SEMESTER | FALSE | SEMESTER |
| FY | FALSE | FY |
| MONTH\_NUM | TRUE | MONTH\_NUM |
| QUARTER\_NUM | TRUE | QUARTER\_NUM |

##### PROFIT\_CENTER\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| PROFIT\_CENTER\_ID | FALSE | PROFIT\_CENTER\_ID |
| PROFIT\_CENTER | FALSE | PROFIT\_CENTER\_NAME |

##### GEOGRAPHY\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| GEO\_ID | TRUE | GEO\_ID |
| CORP\_REP\_GEO\_ID | FALSE | REP\_GEO\_ID |
| CORP\_REPORTING\_MARKET | FALSE | CORP\_REPORTING\_MARKET |
| MIN\_GEO\_ID | TRUE | MIN\_GEO\_ID |
| CORP\_MINOR\_MARKET | FALSE | CORP\_MINOR\_MARKET |

##### CORP\_CUSTOMER\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| CUSTOMER\_ID | TRUE | CUSTOMER\_ID |
| CORP\_CUST\_L9\_ID | FALSE | CORP\_CUST\_L9\_ID |
| CORP\_CUST\_L9 | FALSE | CORP\_CUST\_L9 |

##### MANUAL\_CUSTOMER\_MAPPING

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| CUSTOMER\_MAPPING\_KEY | TRUE | CUSTOMER\_MAPPING\_KEY |
| REPORTING\_CUSTOMER\_ID | FALSE | REPORTING\_CUSTOMER\_ID |
| REPORTING\_CUSTOMER | FALSE | REPORTING\_CUSTOMER\_NAME |
| REGIONAL\_CHANNEL | FALSE | REGIONAL\_CHANNEL |
| SMO | FALSE | SMO |
| CORP\_CUSTOMER\_L9\_ID | FALSE | CORP\_CUST\_L9\_ID |
| CORP\_CUSTOMER\_L9 | FALSE | CORP\_CUST\_L9 |
| CUST\_KEY | TRUE | CUST\_KEY |
| CUSTOMER\_TEAM | FALSE | CUSTOMER\_TEAM |
| CUST\_SUB\_TEAM\_NAME | FALSE | CUST\_SUB\_TEAM\_NAME |
| REGIONAL\_SUB\_CHANNEL | FALSE | REGIONAL\_SUB\_CHANNEL |
| REGIONAL\_MACRO\_CHANNEL | FALSE | REGIONAL\_MACRO\_CHANNEL |
| REGIONAL\_CUSTOMER\_NAME | FALSE | REGIONAL\_CUSTOMER\_NAME |
| SAP\_CUSTOMER\_GROUP\_ID | FALSE | SAP\_CUSTOMER\_GROUP\_ID |
| CUSTOMER\_CHANNEL | FALSE | CUSTOMER\_CHANNEL |
| CUSTOMER\_SUB\_CHANNEL | FALSE | CUSTOMER\_SUB\_CHANNEL |
| CUSTOMER\_NAME | FALSE | CUSTOMER\_NAME |
| ORG\_AREA\_NAME | FALSE | ORG\_AREA\_NAME |

##### Hidden Tables

###### CURRENCY\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| CURRENCY\_ID | FALSE | CURRENCY\_ID |
| CURRENCY\_NAME | FALSE | CURRENCY\_NAME |

###### RLS\_SECURITY

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| rls\_key | FALSE | rls\_key |
| group\_genie | FALSE | group\_genie |

###### TIME\_DIM\_ROLLING

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | FALSE | TIME\_ID |
| time\_id\_ya | FALSE | time\_id\_ya |

#### Relationship diagram

A computer screen shot of a computer

Description automatically generated

#### Calculated KPI’s

Following is the attached document with detailed information on each calculated KPI.



### LA\_CNGC\_Market

* Data Source: SQL/cngc-sqlserver database windows net;cngc-sqldb
* Refresh Mode: Clear 🡪 Process Full
* Refreshed via: Manually triggered via ADF

#### List of Datasets used.

|  |  |
| --- | --- |
| **Table Name** | **Hidden** |
| CNGC\_LA\_OUTPUT\_RSMO\_FACT | No |
| CNGC\_LA\_SD\_FACT | No |
| CORP\_CUSTOMER\_DIM | No |
| currency\_dim | Yes |
| GEOGRAPHY\_DIM | No |
| MANNUAL\_CUSTOMER\_MAPPING | No |
| PRODUCT\_DIM | No |
| PROFIT\_CENTER\_DIM | No |
| rls\_security | Yes |
| TIME\_DIM | No |
| time\_dim\_rolling | Yes |

#### Details of columns for each dataset.

##### TIME\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | sourceColumn |
| TIME\_ID | TRUE | TIME\_ID |
| MONTH | FALSE | MONTH |
| QUARTER | FALSE | QUARTER |
| SEMESTER | FALSE | SEMESTER |
| FY | FALSE | FY |
| MONTH\_NUM | TRUE | MONTH\_NUM |
| QUARTER\_NUM | TRUE | QUARTER\_NUM |

##### PROFIT\_CENTER\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| PROFIT\_CENTER\_ID | FALSE | PROFIT\_CENTER\_ID |
| PROFIT\_CENTER | FALSE | PROFIT\_CENTER\_NAME |

##### GEOGRAPHY\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| GEO\_ID | TRUE | GEO\_ID |
| CORP\_REP\_GEO\_ID | FALSE | REP\_GEO\_ID |
| CORP\_REPORTING\_MARKET | FALSE | CORP\_REPORTING\_MARKET |
| MIN\_GEO\_ID | TRUE | MIN\_GEO\_ID |
| CORP\_MINOR\_MARKET | FALSE | CORP\_MINOR\_MARKET |

##### PRODUCT\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| PRODUCT\_ID | TRUE | PRODUCT\_ID |
| FPC | FALSE | FPC\_NAME |
| BRAND\_FORM\_ID | FALSE | BRAND\_FORM\_ID |
| BRAND\_ID | FALSE | BRAND\_ID |
| CATEGORY\_ID | FALSE | CATEGORY\_ID |
| SUBSECTOR\_ID | FALSE | SUB\_SECTOR\_ID |
| BRAND\_FORM | FALSE | BRAND\_FORM\_NAME |
| BRAND | FALSE | BRAND\_NAME |
| CATEGORY | FALSE | CATEGORY\_NAME |
| FPC\_ID | FALSE | FPC\_ID |
| SUBSECTOR | FALSE | SUB\_SECTOR\_NAME |

##### CNGC\_LA\_OUTPUT\_RSMO\_FACT

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | TRUE | TIME\_ID |
| CUSTOMER\_ID | TRUE | CUSTOMER\_ID |
| GEO\_ID | TRUE | GEO\_ID |
| PRODUCT\_ID | TRUE | PRODUCT\_ID |
| CURRENCY\_ID | TRUE | CURRENCY\_ID |
| PROFIT\_CENTER\_ID | TRUE | PROFIT\_CENTER\_ID |
| NIV\_AMT | TRUE | NIV\_AMT |
| NIV2\_AMT | TRUE | NIV2\_AMT |
| REPORTING\_CUSTOMER\_ID | TRUE | REPORTING\_CUSTOMER\_ID |
| REPORTING\_CUSTOMER | TRUE | REPORTING\_CUSTOMER\_NAME |
| REGIONAL\_CHANNEL | TRUE | REGIONAL\_CHANNEL |
| CUSTOMER\_CHANNEL | FALSE | CUSTOMER\_CHANNEL |
| CUSTOMER\_SUB\_CHANNEL | FALSE | CUSTOMER\_SUB\_CHANNEL |
| CUST\_KEY | TRUE | CUST\_KEY |
| PROD\_KEY | TRUE | PROD\_KEY |
| RLS\_KEY | TRUE | RLS\_KEY |
| NIT\_AMT\_LC | TRUE | NIT\_AMT\_LC |
| NIT\_AMT\_RUSD | TRUE | NIT\_AMT\_RUSD |
| NIT\_AMT\_CUSD | TRUE | NIT\_AMT\_CUSD |
| TDC\_AMT\_LC | TRUE | TDC\_AMT\_LC |
| TDC\_AMT\_RUSD | TRUE | TDC\_AMT\_RUSD |
| TDC\_AMT\_CUSD | TRUE | TDC\_AMT\_CUSD |
| NSRD\_DAMAGE\_RETURNS\_AMT\_LC | TRUE | NSRD\_DAMAGE\_RETURNS\_AMT\_LC |
| NSRD\_DAMAGE\_RETURNS\_AMT\_RUSD | TRUE | NSRD\_DAMAGE\_RETURNS\_AMT\_RUSD |
| NSRD\_DAMAGE\_RETURNS\_AMT\_CUSD | TRUE | NSRD\_DAMAGE\_RETURNS\_AMT\_CUSD |
| NSRD\_PRICE\_ADJUSTMENTS\_AMT\_LC | TRUE | NSRD\_PRICE\_ADJUSTMENTS\_AMT\_LC |
| NSRD\_PRICE\_ADJUSTMENTS\_AMT\_RUSD | TRUE | NSRD\_PRICE\_ADJUSTMENTS\_AMT\_RUSD |
| NSRD\_PRICE\_ADJUSTMENTS\_AMT\_CUSD | TRUE | NSRD\_PRICE\_ADJUSTMENTS\_AMT\_CUSD |
| NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_LC | TRUE | NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_LC |
| NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_RUSD | TRUE | NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_RUSD |
| NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_CUSD | TRUE | NSRD\_DISTRIBUTOR\_DISCOUNTS\_AMT\_CUSD |
| NSRD\_OTHERS\_AMT\_LC | TRUE | NSRD\_OTHERS\_AMT\_LC |
| NSRD\_OTHERS\_AMT\_RUSD | TRUE | NSRD\_OTHERS\_AMT\_RUSD |
| NSRD\_OTHERS\_AMT\_CUSD | TRUE | NSRD\_OTHERS\_AMT\_CUSD |
| NSRD\_PROMPT\_PAYMENT\_AMT\_LC | TRUE | NSRD\_PROMPT\_PAYMENT\_AMT\_LC |
| NSRD\_PROMPT\_PAYMENT\_AMT\_RUSD | TRUE | NSRD\_PROMPT\_PAYMENT\_AMT\_RUSD |
| NSRD\_PROMPT\_PAYMENT\_AMT\_CUSD | TRUE | NSRD\_PROMPT\_PAYMENT\_AMT\_CUSD |
| NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_LC | TRUE | NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_LC |
| NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_RUSD | TRUE | NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_RUSD |
| NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_CUSD | TRUE | NSRD\_LOGISTIC\_DISCOUNTS\_AMT\_CUSD |
| NSRD\_WRITE\_OFF\_AMT\_LC | TRUE | NSRD\_WRITE\_OFF\_AMT\_LC |
| NSRD\_WRITE\_OFF\_AMT\_RUSD | TRUE | NSRD\_WRITE\_OFF\_AMT\_RUSD |
| NSRD\_WRITE\_OFF\_AMT\_CUSD | TRUE | NSRD\_WRITE\_OFF\_AMT\_CUSD |
| SAP\_GROSS\_SALES\_AMT\_LC | TRUE | SAP\_GROSS\_SALES\_AMT\_LC |
| SAP\_GROSS\_SALES\_AMT\_RUSD | TRUE | SAP\_GROSS\_SALES\_AMT\_RUSD |
| SAP\_GROSS\_SALES\_AMT\_CUSD | TRUE | SAP\_GROSS\_SALES\_AMT\_CUSD |
| CAT\_NSRD\_TIE\_OUT\_LC | TRUE | CAT\_NSRD\_TIE\_OUT\_LC |
| CAT\_SD\_TIE\_OUT\_LC | TRUE | CAT\_SD\_TIE\_OUT\_LC |
| CAT\_TDC\_TIE\_OUT\_LC | TRUE | CAT\_TDC\_TIE\_OUT\_LC |
| TIE\_OUT\_NSRD\_LC | TRUE | TIE\_OUT\_NSRD\_LC |
| TIE\_OUT\_NSRD\_RUSD | TRUE | TIE\_OUT\_NSRD\_RUSD |
| TIE\_OUT\_NSRD\_CUSD | TRUE | TIE\_OUT\_NSRD\_CUSD |
| TIE\_OUT\_SD\_LC | TRUE | TIE\_OUT\_SD\_LC |
| TIE\_OUT\_SD\_RUSD | TRUE | TIE\_OUT\_SD\_RUSD |
| TIE\_OUT\_SD\_CUSD | TRUE | TIE\_OUT\_SD\_CUSD |
| TIE\_OUT\_TDC\_LC | TRUE | TIE\_OUT\_TDC\_LC |
| TIE\_OUT\_TDC\_RUSD | TRUE | TIE\_OUT\_TDC\_RUSD |
| TIE\_OUT\_TDC\_CUSD | TRUE | TIE\_OUT\_TDC\_CUSD |
| NOS\_C\_LC | TRUE | NOS\_LC |
| NOS\_C\_RUSD | TRUE | NOS\_RUSD |
| NOS\_C\_CUSD | TRUE | NOS\_CUSD |
| GC\_C\_LC | TRUE | GC\_LC |
| GC\_C\_RUSD | TRUE | GC\_RUSD |
| GC\_C\_CUSD | TRUE | GC\_CUSD |
| NSRD\_TOTAL\_C\_LC | TRUE | NSRD\_TOTAL\_LC |
| NSRD\_TOTAL\_C\_CUSD | TRUE | NSRD\_TOTAL\_CUSD |
| NSRD\_TOTAL\_C\_RUSD | TRUE | NSRD\_TOTAL\_RUSD |
| SD\_TOTAL\_C\_LC | TRUE | SD\_TOTAL\_LC |
| SD\_TOTAL\_C\_CUSD | TRUE | SD\_TOTAL\_CUSD |
| SD\_TOTAL\_C\_RUSD | TRUE | SD\_TOTAL\_RUSD |
| SMO | TRUE | SMO |
| GIV\_C\_LC | TRUE | GIV\_LC |
| GIV\_C\_RUSD | TRUE | GIV\_RUSD |
| GIV\_C\_CUSD | TRUE | GIV\_CUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_LC | TRUE | NSRD\_MANUAL\_INPUT\_AMT\_LC |
| NSRD\_MANUAL\_INPUT\_AMT\_RUSD | TRUE | NSRD\_MANUAL\_INPUT\_AMT\_RUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_CUSD | TRUE | NSRD\_MANUAL\_INPUT\_AMT\_CUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_YA\_LC | TRUE | NSRD\_MANUAL\_INPUT\_AMT\_YA\_LC |
| NSRD\_MANUAL\_INPUT\_AMT\_YA\_RUSD | TRUE | NSRD\_MANUAL\_INPUT\_AMT\_YA\_RUSD |
| NSRD\_MANUAL\_INPUT\_AMT\_YA\_CUSD | TRUE | NSRD\_MANUAL\_INPUT\_AMT\_YA\_CUSD |
| VOLUME\_C | TRUE | VOLUME |
| CUSTOMER\_MAPPING\_KEY | TRUE | CUSTOMER\_MAPPING\_KEY |
| ATTR29 | TRUE | ATTR29 |
| ATTR17 | TRUE | ATTR17 |
| ATTR19 | TRUE | ATTR19 |
| ATTR18 | TRUE | ATTR18 |
| ATTR2 | TRUE | ATTR2 |
| ATTR3 | TRUE | ATTR3 |
| ATTR1 | TRUE | ATTR1 |
| REGIONAL\_SUB\_CHANNEL | FALSE | REGIONAL\_SUB\_CHANNEL |
| REGIONAL\_MACRO\_CHANNEL | FALSE | REGIONAL\_MACRO\_CHANNEL |
| REGIONAL\_CUSTOMER\_NAME | FALSE | REGIONAL\_CUSTOMER\_NAME |
| SAP\_CUSTOMER\_GROUP\_ID | FALSE | SAP\_CUSTOMER\_GROUP\_ID |
| CUSTOMER\_NAME | FALSE | CUSTOMER\_NAME |
| ORG\_AREA\_NAME | FALSE | ORG\_AREA\_NAME |

##### CNGC LA\_SD\_FACT

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | TRUE | TIME\_ID |
| CUSTOMER\_ID | TRUE | CUSTOMER\_ID |
| GEO\_ID | TRUE | GEO\_ID |
| PRODUCT\_ID | TRUE | PRODUCT\_ID |
| CURRENCY\_ID | TRUE | CURRENCY\_ID |
| RLS\_KEY | TRUE | RLS\_KEY |
| SD\_TPR\_AMT\_LC | TRUE | SD\_TPR\_AMT\_LC |
| SD\_TPR\_C\_RUSD | TRUE | SD\_TPR\_RUSD |
| SD\_TPR\_C\_CUSD | TRUE | SD\_TPR\_CUSD |
| SD\_LIVE\_RATES\_AMT\_LC | TRUE | SD\_LIVE\_RATES\_AMT\_LC |
| SD\_LIVE\_RATES\_C\_RUSD | TRUE | SD\_LIVE\_RATES\_RUSD |
| SD\_LIVE\_RATES\_C\_CUSD | TRUE | SD\_LIVE\_RATES\_CUSD |
| SD\_FIXED\_BUDGET\_AMT\_LC | TRUE | SD\_FIXED\_BUDGET\_AMT\_LC |
| SD\_FIXED\_BUDGET\_C\_RUSD | TRUE | SD\_FIXED\_BUDGET\_RUSD |
| SD\_FIXED\_BUDGET\_C\_CUSD | TRUE | SD\_FIXED\_BUDGET\_CUSD |
| SD\_MANUAL\_INPUT\_C\_LC | TRUE | SD\_MANUAL\_INPUT\_LC |
| SD\_MANUAL\_INPUT\_C\_RUSD | TRUE | SD\_MANUAL\_INPUT\_RUSD |
| SD\_MANUAL\_INPUT\_C\_CUSD | TRUE | SD\_MANUAL\_INPUT\_CUSD |
| CUSTOMER\_MAPPING\_KEY | TRUE | CUSTOMER\_MAPPING\_KEY |
| PROFIT\_CENTER\_ID | TRUE | PROFIT\_CENTER\_ID |

##### CORP\_CUSTOMER\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| CUSTOMER\_ID | TRUE | CUSTOMER\_ID |
| CORP\_CUST\_L9\_ID | FALSE | CORP\_CUST\_L9\_ID |
| CORP\_CUST\_L9 | FALSE | CORP\_CUST\_L9 |

##### MANUAL\_CUSTOMER\_MAPPING

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| CUSTOMER\_MAPPING\_KEY | TRUE | CUSTOMER\_MAPPING\_KEY |
| REPORTING\_CUSTOMER\_ID | FALSE | REPORTING\_CUSTOMER\_ID |
| REPORTING\_CUSTOMER | FALSE | REPORTING\_CUSTOMER\_NAME |
| REGIONAL\_CHANNEL | FALSE | REGIONAL\_CHANNEL |
| CUSTOMER\_CHANNEL | FALSE | CUSTOMER\_CHANNEL |
| CUSTOMER\_SUB\_CHANNEL | FALSE | CUSTOMER\_SUB\_CHANNEL |
| SMO | FALSE | SMO |
| CORP\_CUSTOMER\_L9\_ID | FALSE | CORP\_CUST\_L9\_ID |
| CORP\_CUSTOMER\_L9 | FALSE | CORP\_CUST\_L9 |
| CUST\_KEY | TRUE | CUST\_KEY |
| CUSTOMER\_TEAM | FALSE | CUSTOMER\_TEAM |
| CUST\_SUB\_TEAM\_NAME | FALSE | CUST\_SUB\_TEAM\_NAME |
| CUSTOMER\_NAME | FALSE | CUSTOMER\_NAME |
| ORG\_AREA\_NAME | FALSE | ORG\_AREA\_NAME |
| REGIONAL\_SUB\_CHANNEL | FALSE | REGIONAL\_SUB\_CHANNEL |
| REGIONAL\_MACRO\_CHANNEL | FALSE | REGIONAL\_MACRO\_CHANNEL |
| REGIONAL\_CUSTOMER\_NAME | FALSE | REGIONAL\_CUSTOMER\_NAME |
| SAP\_CUSTOMER\_GROUP\_ID | FALSE | SAP\_CUSTOMER\_GROUP\_ID |

##### Hidden Tables

###### CURRENCY\_DIM

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| CURRENCY\_ID | FALSE | CURRENCY\_ID |
| CURRENCY\_NAME | FALSE | CURRENCY\_NAME |

###### RLS\_SECURITY

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| rls\_key | FALSE | rls\_key |
| group\_genie | FALSE | group\_genie |

###### TIME\_DIM\_ROLLING

|  |  |  |
| --- | --- | --- |
| **Name** | **isHidden** | **sourceColumn** |
| TIME\_ID | FALSE | TIME\_ID |
| time\_id\_ya | FALSE | time\_id\_ya |

#### A computer screen shot of a computer Description automatically generatedRelationship diagram

#### Calculated KPI’s

Following is the attached document with detailed information on each calculated KPI.



### LA\_CNGC\_Market\_Archive

* The Market Archive model is an exact replica of the Market model and there is no difference in their configurations.
* Market Archive Model is used when fiscal years are switched (FYE/FYB), to retain the data published before the FY change.