



DPF Console Implementation Document

***Prepared By:***

***LTIMindtree Ltd.***

Document Version/Details: Ver. 1.0/ 25-Jul-2023

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Revised by | Description |
| 1.0 | Jan 20, 2023 | Abhay Saxena | Draft version |
| 1.1 | Nov 8, 2023 | Java, UI, DE Team | Draft Version |
| 2 | Nov 9, 2023 | Rahul Pakhare  Gopi C | Final Version |

Approvers

|  |  |  |
| --- | --- | --- |
| Approver | Role | Date |
|  |  |  |
| Anirban Sarkar | Associate Director |  |
| Andrzej Bernad | P&G |  |
| Ievgenii Poltoratskyi | P&G |  |
| Piotr Janusz | P&G |  |

Table of Contents

[1Introduction 2](#_Toc1701012579)

[1.1 proprietary notice 2](#_Toc789692953)

[1.2 Audience 2](#_Toc1481663543)

[2. Cloud Panel Architecture 3](#_Toc1755557121)

[2.5 Cloud Panel Architecture Diagram 3](#_Toc376172402)

[2.6 Design Approach 4](#_Toc351999329)

[2.7 Contract Save - Flow Diagram 5](#_Toc1993459135)

[2.8 Integration with Unrefined Objects from UI 6](#_Toc512756658)

[2.9 DB Details (Details would change for higher Environments) 10](#_Toc289710484)

[2.10 Storage Account (Details would change for higher Environments) 10](#_Toc1234889950)

[2.11 Resource Group 11](#_Toc1741507648)

[3. Reference Data 11](#_Toc1023425056)

[3.5 Contacts 12](#_Toc1067453352)

[3.6 Measures 25](#_Toc1841945724)

[3.7 Products 28](#_Toc1933331470)

[3.7.1 Data Standards 29](#_Toc1243600190)

[3.7.2 Attribute Hierarchies 36](#_Toc158471211)

[3.7.3 Attribute Values 39](#_Toc46041940)

[3.8 Areas 40](#_Toc1844964079)

[3.8.1 Area Aggregates 40](#_Toc1100650233)

[3.8.2 TP Groups 43](#_Toc1458260586)

[3.8.3 TP Mappings 44](#_Toc1726123561)

[3.9 Time periods 46](#_Toc306058796)

[3.10 Upload Schedule 48](#_Toc1422947800)

[3.11 HHP Mappings: 50](#_Toc379225530)

[3.11.1 MM\_HHP\_DATA\_CNTXT\_MAPNG 51](#_Toc400817008)

[3.11.2 MM\_HHP\_TIME\_MAPNG 54](#_Toc267072537)

[3.11.3 MM\_HHP\_BUYR\_GRP\_CLASS 55](#_Toc186719800)

[3.11.4 MM\_HHP\_GEO\_CODE\_MAPNG 55](#_Toc1910334427)

[3.11.5 MM\_HHP\_ONE 56](#_Toc1841921165)

[3.11.6 MM\_HHP\_TWO 56](#_Toc1139729948)

[3.11.7 MM\_HHP\_THREE 57](#_Toc508916943)

[4. Contracts 57](#_Toc1425418678)

[4.1 Columns Dropdown: 58](#_Toc542027203)

[4.2 Tier 2 Contract Creation: 64](#_Toc1301100247)

[4.2.1 Data Delivery: 65](#_Toc75083345)

[4.2.2 Data Quality Validation: 66](#_Toc1814235651)

[4.2.3 Select File and Assign Columns: 67](#_Toc572745557)

[4.2.4 Time Dimensional Mapping: 68](#_Toc332353819)

[4.2.5 Contract activation: 68](#_Toc193182790)

[4.3 HHP Contract Creation: 69](#_Toc1218992796)

[4.3.1 Measure Multiplicators: 70](#_Toc611862139)

[4.3.2 Select File: 71](#_Toc2082667577)

[4.3.3 Assigned Columns (Mappings): 72](#_Toc272536679)

[4.3.4 Special Data Validation: 73](#_Toc2056722265)

[4.3.5 Contract Activation: 74](#_Toc1301627831)

[4.3.6 Runs Rejection: 74](#_Toc107375126)

[4.4 Tier 1 Contract: 75](#_Toc1650199628)

[4.4.1 Tier-1 Contract Creation: 76](#_Toc1209153657)

[4.4.2 Data Quality and Validation Level: 76](#_Toc1452455340)

[4.4.3 DQ Validation Page: 77](#_Toc1302317027)

[4.4.4 Multiplicators: 78](#_Toc1720756982)

[4.4.5 Contract Activation: 79](#_Toc126930611)

[5. Files 84](#_Toc1096855959)

[5.1 Include rejected files: 85](#_Toc568883158)

[5.2 View 86](#_Toc1359209741)

[5.3 HHP Vendor Button 87](#_Toc877427700)

[5.4 Upload File Button 88](#_Toc1968680333)

[5.5 New Contract 89](#_Toc339050315)

[5.6 Move To Inbox: 91](#_Toc155949890)

[6. Runs 93](#_Toc596055961)

[6.1 Auto Refresh: 93](#_Toc1535866804)

[6.2 Logs: 94](#_Toc1270424358)

[6.3 Action Dropdown: 95](#_Toc488950642)

[6.3.1 Resume: 96](#_Toc558088383)

[6.3.2 Full Reload: 97](#_Toc90794977)

[6.3.3 Continue: 100](#_Toc1986601931)

[6.3.4 Remove: 101](#_Toc238911417)

[6.3.5 Download: 104](#_Toc1907965143)

[6.3.6 View Contract: 105](#_Toc1500002108)

[6.3.7 Edit Contract: 106](#_Toc2104028286)

[6.4 Refresh Button 106](#_Toc1262702372)

[6.5 Clear Filter Button: 107](#_Toc458868632)

[7. Deliveries: 107](#_Toc598934667)

[7.1 Assign File Owner: 108](#_Toc1162071462)

[7.2 Auto Refresh: 109](#_Toc44905133)

[7.3 Actions dropdown button: 109](#_Toc1033922599)

[7.4 Validation Report: 110](#_Toc1620166140)

[7.5 Business Validation: 111](#_Toc1206976685)

[7.6 Approve: 111](#_Toc215078875)

[7.7 Reject: 113](#_Toc253440687)

[7.8 Release: 115](#_Toc1983837638)

[7.9 Refresh Integrated: 115](#_Toc590937410)

[7.10 Full Reload: 115](#_Toc1003403626)

[7.11 Rollback: 116](#_Toc1998159560)

[7.12 Runs: 116](#_Toc1112496318)

[7.13 Download: 117](#_Toc2000218710)

[8. Administration. 119](#_Toc293621382)

[8.1 Chains. 119](#_Toc1253870315)

[8.2 Application Logs. 119](#_Toc2036542280)

[8.3 Security Scopes 120](#_Toc765836411)

[8.4 Source System 124](#_Toc1911206612)

[8.5 Vendors 126](#_Toc914264568)

[8.6 Integrated Contract 129](#_Toc439661064)

[9. Authentication And Authorization: 130](#_Toc1507935305)

[9.1 Authentication: 132](#_Toc1927879588)

[9.1.1 Steps of DPF Console UI Secured Login: 132](#_Toc2017553170)

[9.1.2 Authentication Error: 134](#_Toc1178822099)

[9.1.3 Steps of DPF Console UI Logout: 134](#_Toc1412135767)

[9.2 Authorization: 135](#_Toc178080095)

[9.2.1 User Roles and AD groups: 136](#_Toc314204385)

[9.2.2 DB tables/views used for the authorization: 138](#_Toc29422700)

[9.3 Azure Active Directory App Registration: 139](#_Toc383784234)

[10. Azure Integration 139](#_Toc1342427879)

[10.1 Azure Key Vault Integration: 139](#_Toc1821300824)

[10.1.1 DPF Console application Key vaults: 140](#_Toc590298718)

[10.2 Continuous integration and continuous deployment: 142](#_Toc808135054)

[11. Chain/graph structure 143](#_Toc94712514)

[11.1 DPF\_Tier1\_Atomic TP Main Chain SFF 143](#_Toc1538128426)

[11.1.1 File Parsing 143](#_Toc483664587)

[11.1.2 Nested graph: T1\_SFF Fact Standardization 145](#_Toc14328680)

[11.1.3 Nested graph:T1\_SFF Run Logs Loading 145](#_Toc2115733985)

[11.1.4 Nested graph:T1\_SFF\_Atomic Measure Calculation 145](#_Toc1798975139)

[11.1.5 Nested graph: T1\_SFF Fact Dimension Skid 146](#_Toc45400585)

[11.1.6 Nested graph: T1\_SFF Fact Image Derivation 146](#_Toc2015750821)

[11.1.7 Nested graph: T1\_SFF\_Measure\_log\_Loading 146](#_Toc392117849)

[11.1.8 Nested graph: T1\_SFF Fact Copy 146](#_Toc601073666)

[11.1.9 Nested graph:T1\_SFF Business Data Quality 147](#_Toc220852225)

[11.1.10 Nested graph: T1\_SFF\_Atomic Product Publication 148](#_Toc1258507541)

[11.1.11 Nested graph: T1\_SFF\_Atomic\_Fact\_Publication 148](#_Toc506668216)

[11.2 DPF\_Tier1\_Atomic TP Main Chain 148](#_Toc269725046)

[11.2.1 File Parsing 148](#_Toc1947108723)

[11.2.2 Nested graph: T1\_Product Derivation 149](#_Toc361368065)

[11.2.3 Nested graph: T1\_Market Derivation 150](#_Toc11229498)

[11.2.4 Nested graph: T1\_Fact\_Standardization\_v1 150](#_Toc1261677151)

[11.2.5 Nested graph: T1\_Run\_Log\_Trans 151](#_Toc1382943682)

[11.2.6 Nested graph: T1\_Atomic Measure Calculations 151](#_Toc474358957)

[11.2.7 Nested graph: T1\_Fact Dimension Skid Assigning 151](#_Toc1933001698)

[11.2.8 Nested graph: T1\_Measure\_Log 151](#_Toc400651073)

[11.2.9 Nested graph: T1 Fact Standard to Months 152](#_Toc1467399019)

[11.2.10 Nested graph: T1\_Runs\_log\_Loading 152](#_Toc1910606532)

[11.2.11 Nested graph: T1\_Atomic\_Product\_Publication 152](#_Toc321465662)

[11.2.12 Nested graph: T1\_Atomic\_Fact\_Publication 152](#_Toc1818230766)

[11.3 DPF\_Tier1\_Integrated TP Main Chain 152](#_Toc790707641)

[11.3.1 Nested graph: T1 Integrated Product Transformation\_v1 153](#_Toc1785041509)

[11.3.2 Nested graph: T1 Integrated Market Transformation\_v1 153](#_Toc1566527959)

[11.3.3 Nested graph:T1 Integrated TP Market Association Loading\_v1 153](#_Toc1979450846)

[11.3.4 Nested graph: T1 Integrated TP Product Association Loading\_v1 153](#_Toc1970140011)

[11.3.5 Nested graph: T1 Integrated Fact Transformation\_v1 153](#_Toc790051401)

[11.3.6 Nested Graph: T1 Integrated Market Custom Aggregate Calculation\_v1 154](#_Toc1630852480)

[11.3.7 Nested Graph: T1 Integrated TP Market Publication\_v1 154](#_Toc125442684)

[11.3.8 Nested Graph: T1 Integrated Product IDIM Contract Loading\_v1 154](#_Toc1909227385)

[11.3.9 Nested Graph: T1 Integrated Market IDIM Contract Loading\_v1 154](#_Toc1142186157)

[11.3.10 Nested graph: T1 Integrated Dimension Publication\_v1 154](#_Toc1867645532)

[11.4 DPF\_Tier2\_USA Nielsen fact categ main chain 154](#_Toc1130357728)

[11.4.1 Nested graph: CDL\_TP\_USA Nielsen Fact-Load Ref Tables 155](#_Toc2071149406)

[11.4.2 Nested graph: DPF2CDL\_TradePanel\_MFT 155](#_Toc1530889752)

[11.4.3 Nested graph: CDL\_TP\_USA Nielsen Fact-Load Dimensions 156](#_Toc8120078)

[11.4.4 Nested graph: CDL\_TP\_USA Nielsen Fact-WK Facts 156](#_Toc1305215888)

[11.5 DPF\_Tier2\_USA Nielsen fact item main chain 157](#_Toc1760396039)

[11.5.1 Nested graph: CDL\_TP\_USA Nielsen Fact-Load Ref Tables 157](#_Toc687069827)

[11.5.2 Nested graph: DPF2CDL\_TradePanel\_MFT 157](#_Toc306437671)

[11.5.3 Nested graph: CDL\_TP\_USA Nielsen Fact-Load Dimensions 158](#_Toc408983783)

[11.5.4 Nested graph: CDL\_TP\_USA Nielsen Fact-WK Facts 158](#_Toc303015174)

[11.6 DPF\_Tier2\_USA Nielsen main prod load chain 159](#_Toc116629698)

[11.6.1 Nested graph: CP\_T2\_import\_MM\_PROD\_PRTTN\_XREF 159](#_Toc306305321)

[11.6.2 Nested graph:CP\_create\_CONNECT\_MM\_PROD\_DIM\_VW 159](#_Toc34323604)

[11.6.3 Nested graph: DPF2CDL\_TradePanel\_MFT 160](#_Toc1922219368)

[12. HHP-Preprocessing 161](#_Toc1502031844)

[12.1 Nested graph: HHP\_AtomicProduct 161](#_Toc2126661993)

[12.2 Nested graph: HHP\_AtomicMarket 162](#_Toc401110360)

[12.3 Nested graph: HHP\_AtomicBuyerGroup 162](#_Toc557323342)

[12.4 Nested graph: HHP\_AtomicFact 162](#_Toc581875717)

[12.5 Nested graph: HHP\_Intgtd\_Layer\_Publish 163](#_Toc636656841)

[13. Glossary 165](#_Toc357635751)

[14. ORM Mappings and DB Diagrams 166](#_Toc1042052399)

[15. References 166](#_Toc582139777)

[16. FAQ 166](#_Toc1261015189)

# 1 Introduction

## proprietary notice

This document contains confidential application information about DPF application belonging to P&G company.

## Audience

The primary audience for this document is Application support & maintenance team, Application Architects, Domain Experts, and other points of contacts.

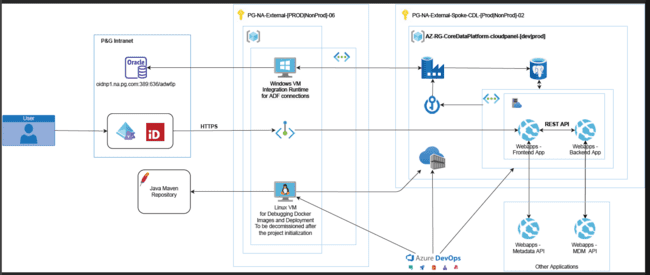
## 

# Cloud Panel Architecture

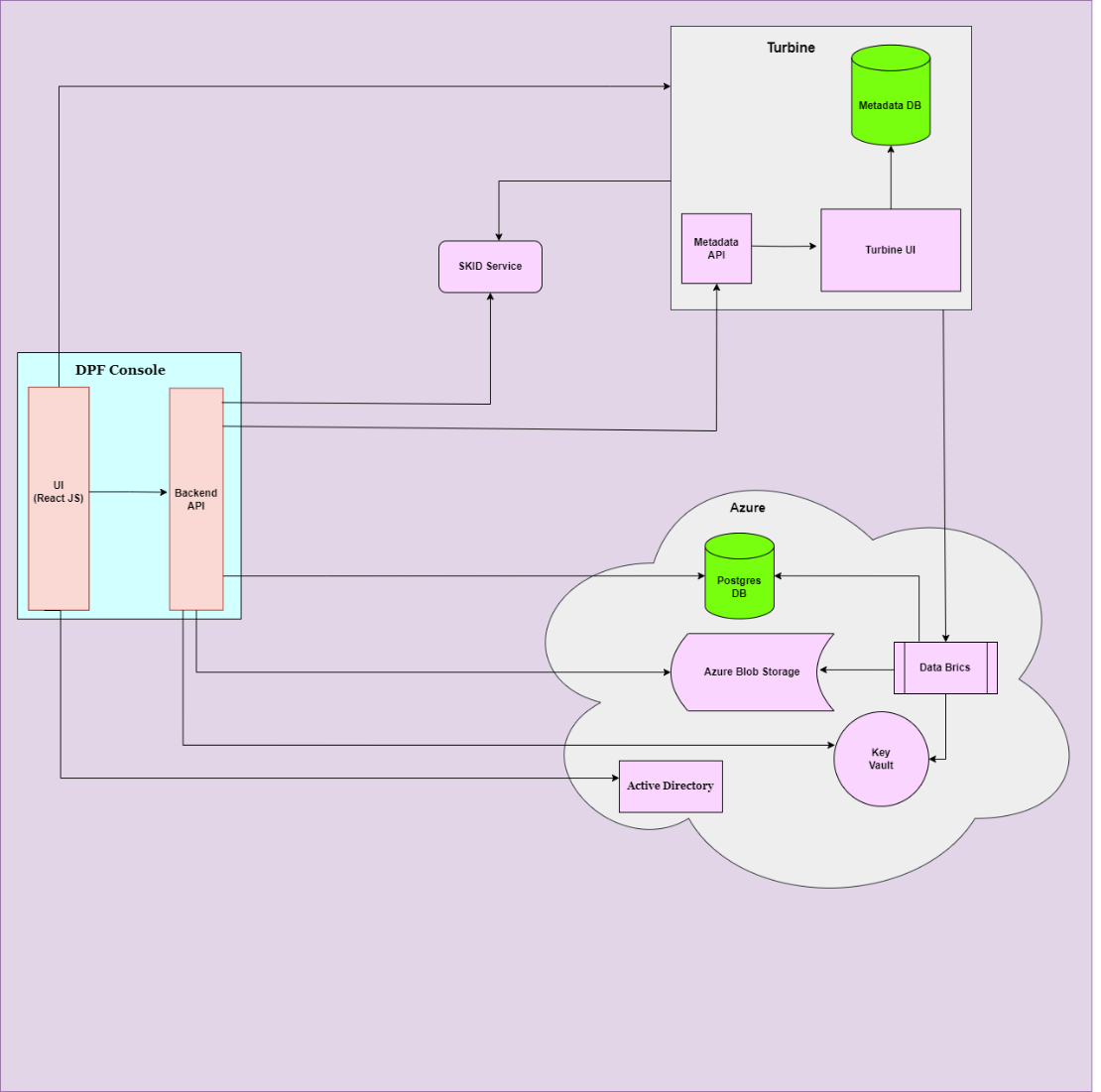
## Cloud Panel Architecture Diagram

Below is the given Architectural diagram for DPF2CDL and the Azure services that we are utilizing.

* Azure Web Apps
* Azure Postgres
* Azure Key Vaults
* Azure ADF
* Azure DevOps



## Interface Diagram



UI application interact with turbine to show the logs of the run.

UI application communicates to the active directory for authentication.

Backend API interacts with Postgres to store & fetch the data from tables

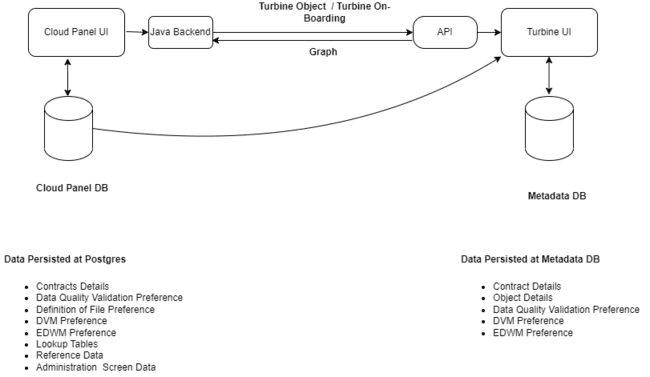
Backend API interacts with Azure Blob Storage to access the files uploaded by vendors.

Backend API interacts with Turbine to create objects & onboardings via metadata API

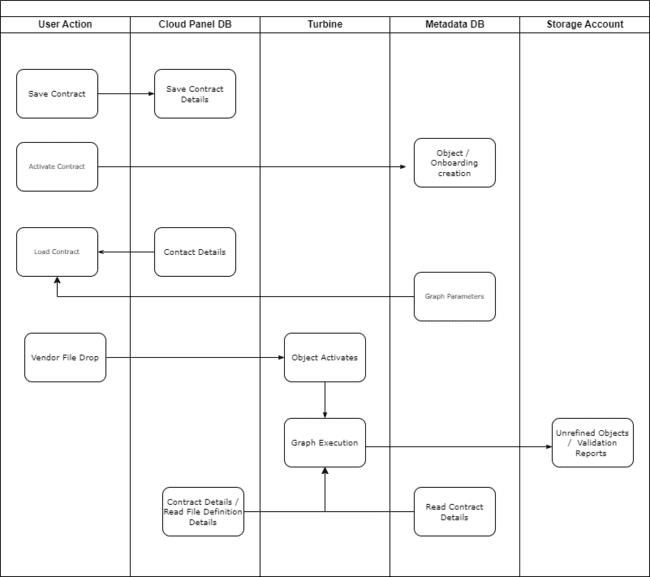
Backend API call SKID service to generate keys.

Databricks uses Azure KeyVault for accessing PostgreSQL database credentials and Managed Service Identity secrets to Mount Blob storages.

## Design Approach



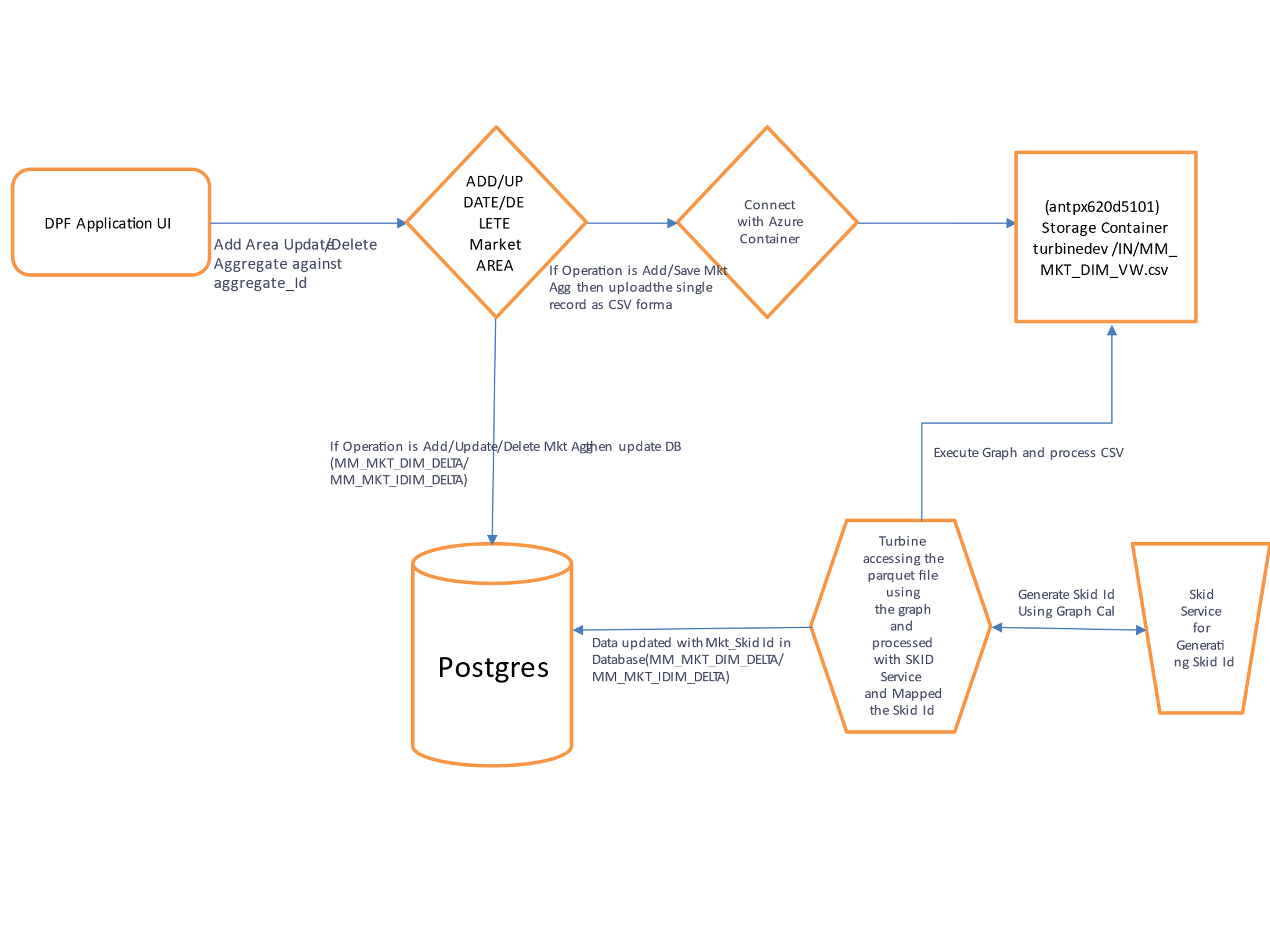
## Contract Save - Flow Diagram



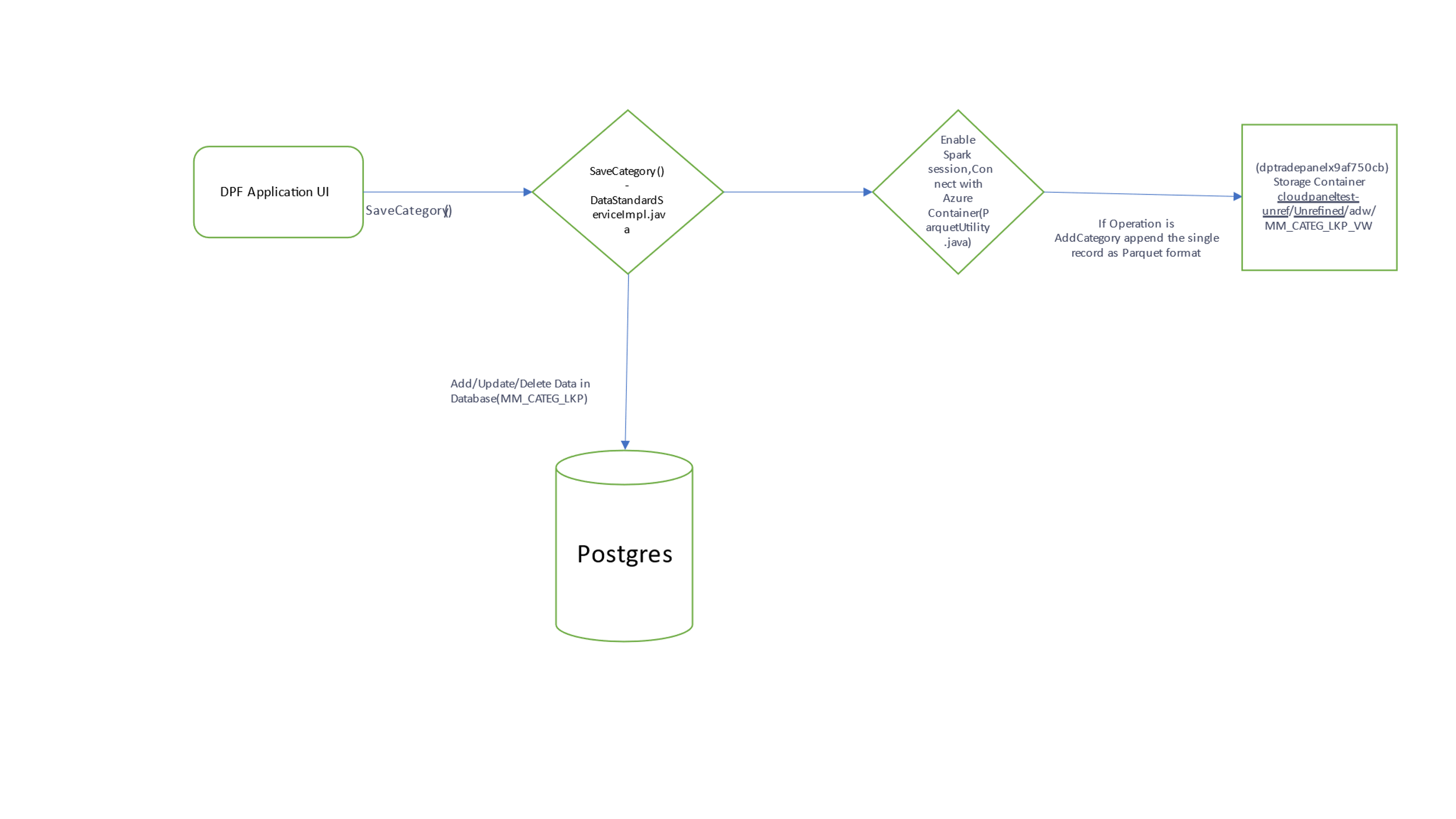
## Integration with Unrefined Objects from UI

Reference Data UI found to be interacting with Unrefined Objects. As Unrefined objects are in Parquet files of Blob storage. We had to create spark session from Java and perform operation on Unrefined objects. Below given are flow diagram explains the flow.

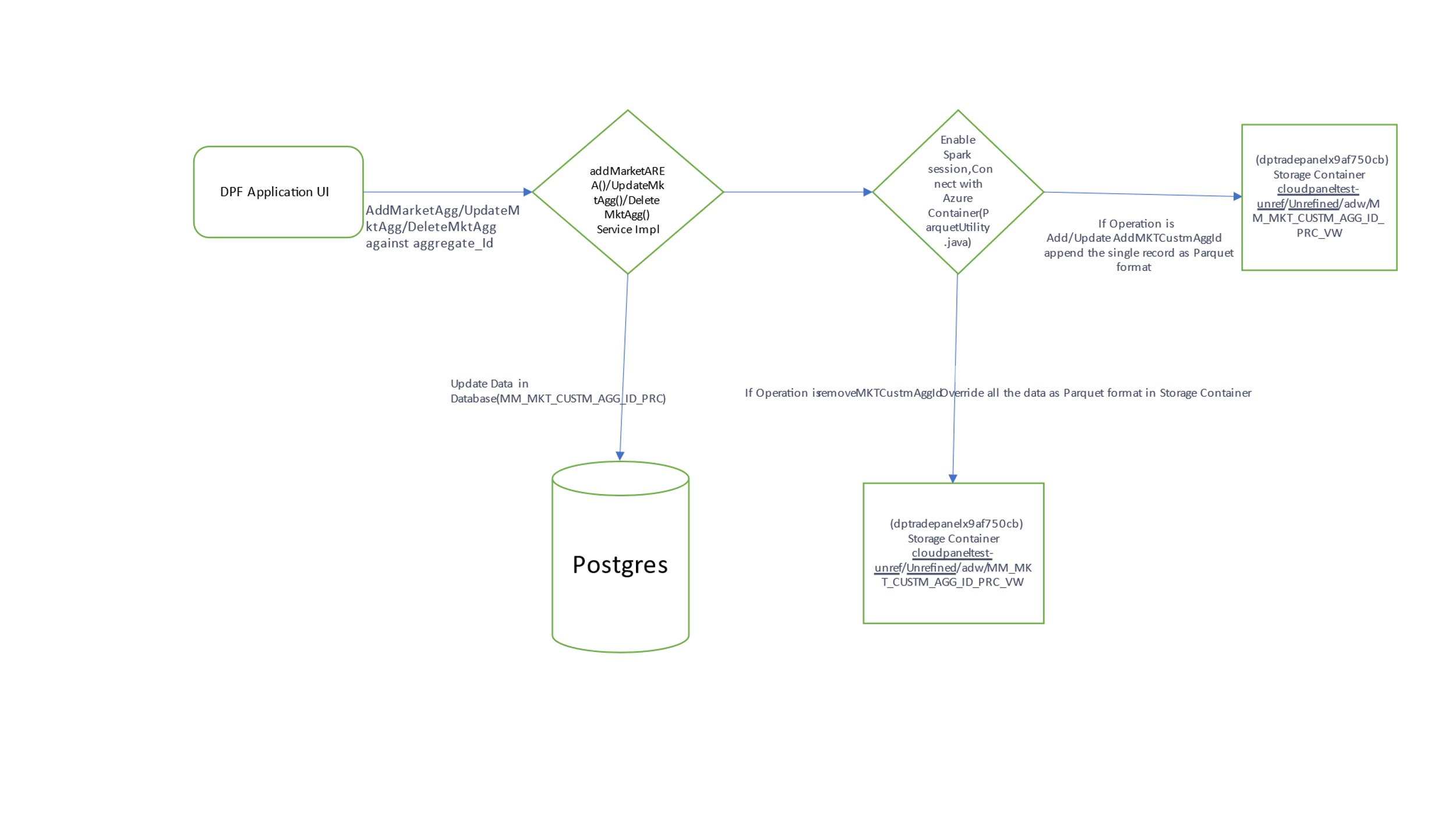
1. ADD MARKET AREA updates MKT\_DIM\_VW

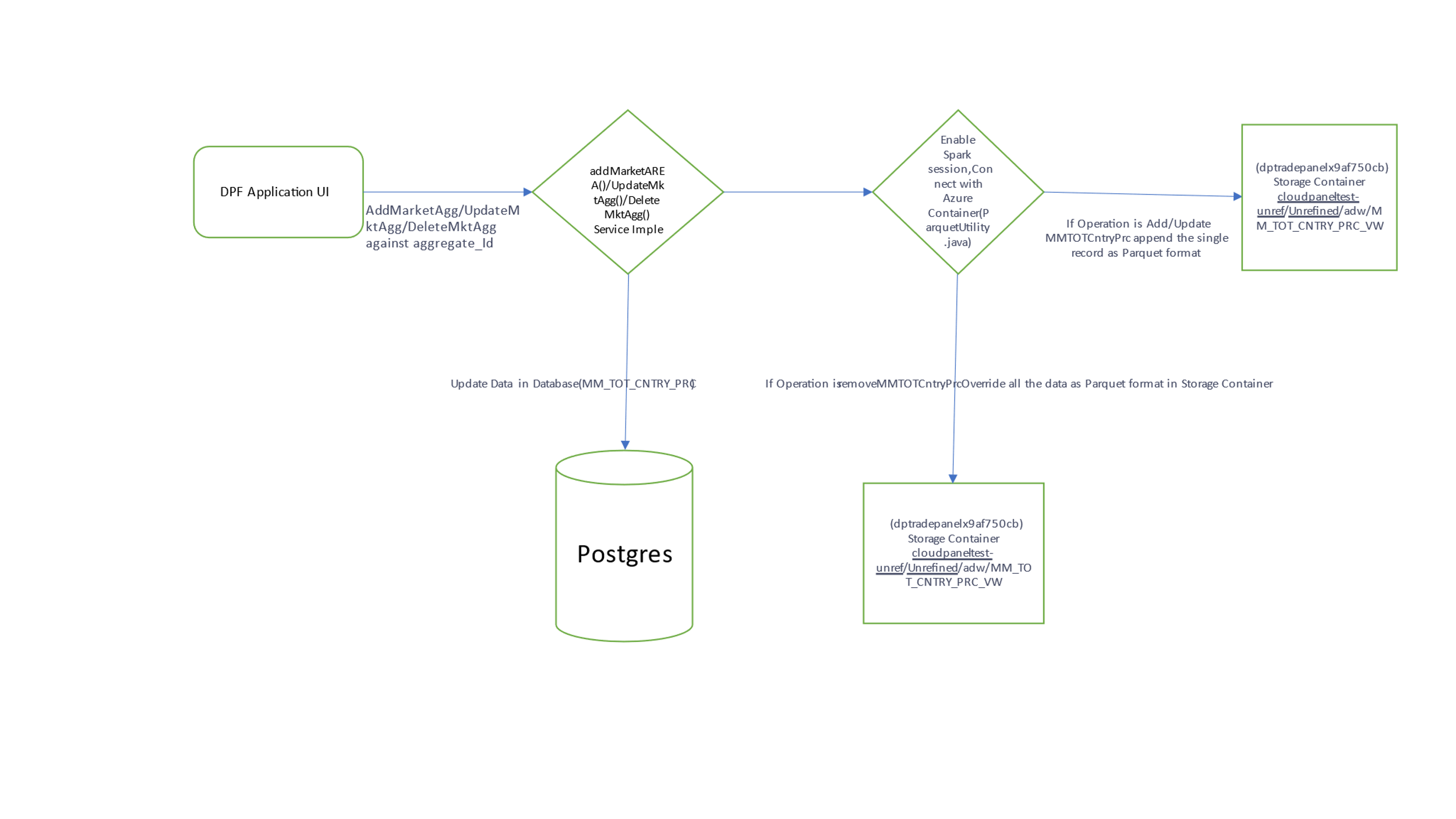


1. Save Category Updates MM\_CATEG\_LKP

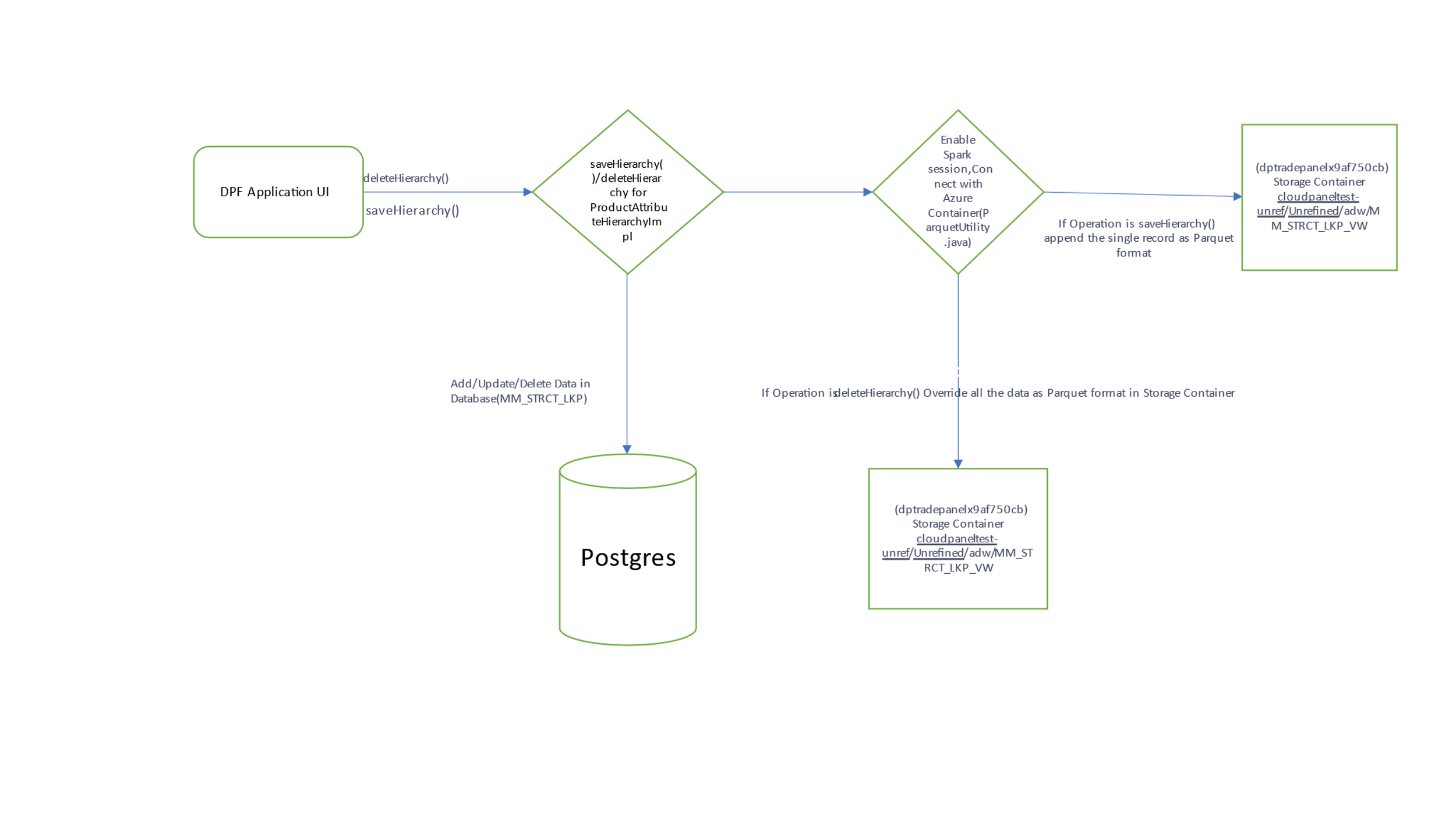


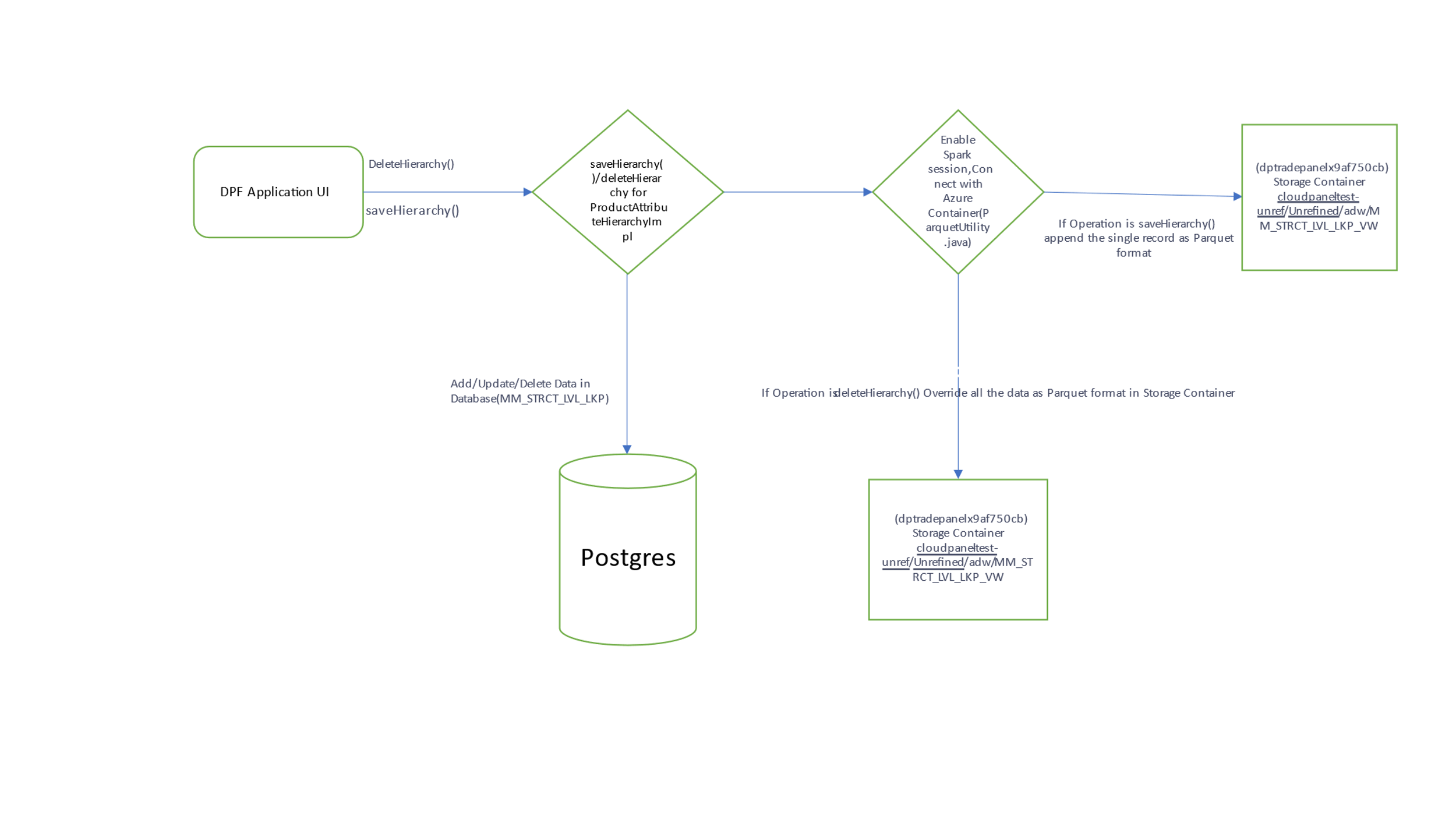
1. ADD / Delete / Update MARKET AREA updated MM\_MKT\_CUSTM\_AGG\_ID\_PRC\_VW and MM\_TOT\_CNTRY\_PRC\_VW





1. Save / Delete Hierarchy updates MM\_STRCT\_LVL\_LKP and MM\_STRCT\_LKP





## DB Details (Details would change for higher Environments)

**SpringBoot Properties**

spring.datasource.driver-class-name=org.postgresql.Driver

spring.datasource.url=jdbc:postgresql://server-postgredb-dev-cloudpanel-1.postgres.database.azure.com/cloudpanel-db

spring.datasource.username=pgadmin@server-postgredb-dev-cloudpanel-1

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL10Dialect

## Storage Account (Details would change for higher Environments)

|  |  |
| --- | --- |
| Unrefined Object Storage Account | dptradepanelx9af750cb |
| Container Name | unrefined |
| Folder Path | /cloudpanel-test-unref/Unrefined/adw/ |

|  |  |
| --- | --- |
| Unrefined Object Storage Account | antpx620d5101 |
| Container Name | turbinev1 |
| Folder Path | /IN/ |

## Resource Group

Below given are different Resource Group being provisioned Azure Resource Groups.

|  |  |  |  |
| --- | --- | --- | --- |
| **Env.** | **Azure Resource Grp** | **Usage** | **Comments** |
| Dev | CoreDataPlatform-CloudPanel-Dev-NP | for all dev work and components | Storage accounts still be in Trade Panel RGs. No VMs allowed in this RG |
| Dev | CoreDataPlatform-CloudPanel-VMonly-NP | for VMs only | If VM is needed, please provision it there |
| QA(UAT) | CoreDataPlatform-CloudPanel-QA-NP | for UAT phases | Environments where users can test |
| PROD | CoreDataPlatform-CloudPanel-Main-PROD | Prod RG for go live | For Go-Live phase |

**PostgreSQL DB Objects**

****

****

**ORM Mapping**



# Reference Data

Ref data deals with Tier 2 data. i.e., Unstructured data from trade panel (vendor collects data from traders of P&G product) or household panel (third party vendor collects information about P&G product directly from customers)

Ref Data in DPF console directs to various other pages by using multi drop down. All screens mentioned below are part of Ref Data. Ref data UI connects with vendors directly to update the details they have given to P&G (working, not working, archived, failed).

Ref Data -> Contacts

Ref Data -> Measures

Ref Data -> Products -> Data Standards

Ref Data -> Products -> Attribute Hierarchies

Ref Data -> Products -> Attribute Values

Ref Data -> Areas -> Aggregates

Ref Data -> Areas -> TP Groups

Ref Data -> Areas -> TP Mappings

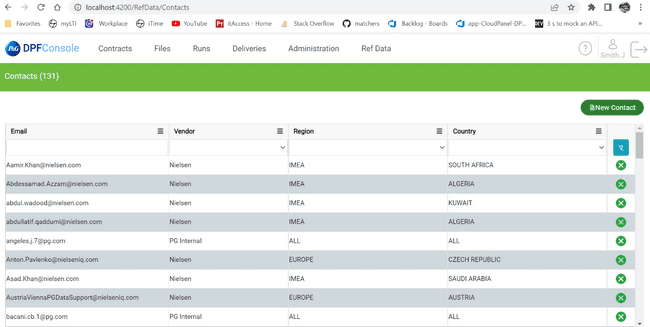
Ref data -> Time Periods

Ref Data -> Upload Schedule

Ref Data -> HHP Mappings

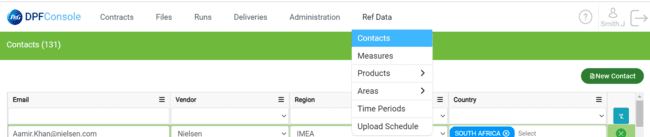
## Contacts

Contact screen deals with the vendor details - email, vendor, region, country. These details help P&G to connect with them for information regarding tier 2 data given by them. Vendors can upload details of the product using contact’s screen.



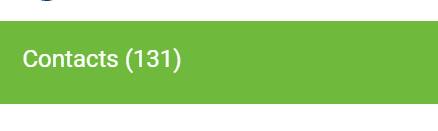
Menu

To access the contacts in menu “Ref Data” Drop down, select Contacts.



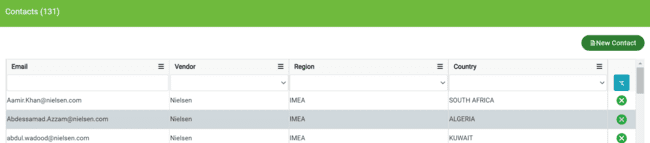
Title

Title of the page “Contacts” with the count of entities in () bracket.



Grid

Grid with column values email, vendor, region, country.



Features:

Edit:

Edit contact is a feature enabled by double clicking on the fields like email, vendor, region, country to change the details or modifying purpose. Changes apply after “enter” button.

Table

Description automatically generated

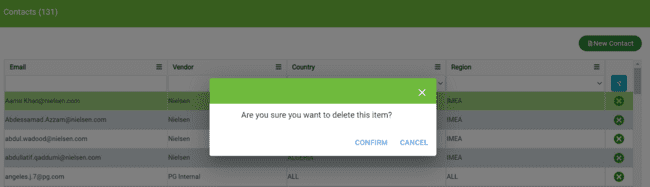
Delete Contact

Delete contact is row level feature located at right side of the row with cross icon. It is used to delete particular row contact.

Table

Description automatically generated

If user tries to delete the data, it pops up message for confirmation as below picture.

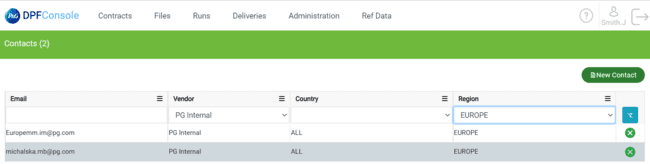


Clear Filter

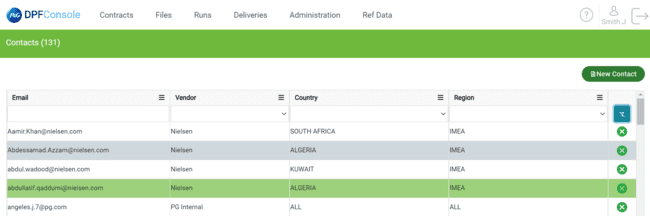
Filter is at top right located below the “New Contact.” It’s used to apply filters on data.

Graphical user interface, application

Description automatically generated

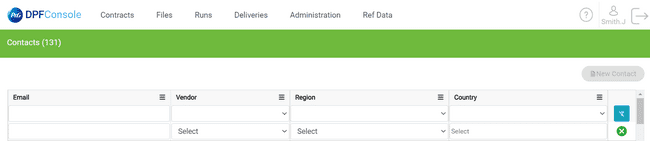


After clearing the filter, the complete data will be displayed.



Add Contact

This page enables user to add a contact using “New Contact” button. That creates a row in Grid. Users can put contact details in the newly added row. To Save User need to press the “Enter” Button.



**Infinite scroll**

It is the feature that deals with scrolling of grid until last entry of data.

**Table

Description automatically generated**

****

**Column options**

By using this feature user can apply conditions for respective column as ascending, descending, configure sort, columns to be there in the grid.

**Ascending & Descending**

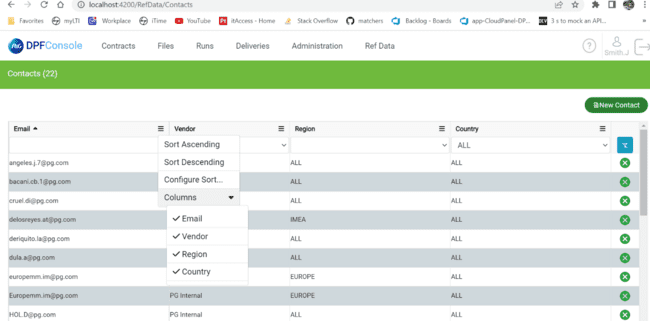
Filtering out the values of user in the grid with ascending, descending order impacts the other columns value with same.

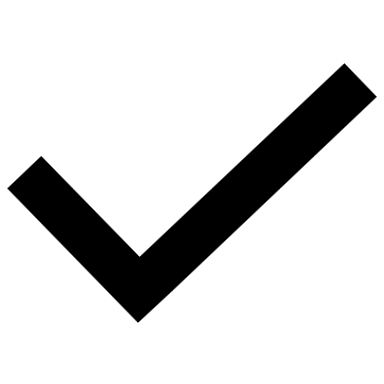
Graphical user interface, table

Description automatically generated

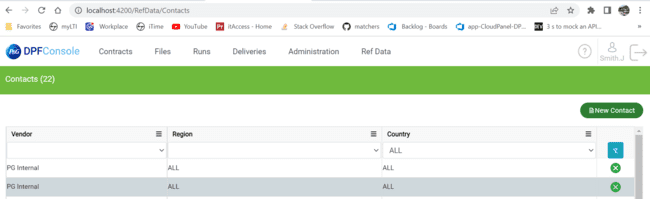
**Show & Hide Columns**

Under column options there is a option called columns which can be used to hide & show columns in the grid



****

by using symbol user can hide the columns. For sample refer below “screenshot”.

****

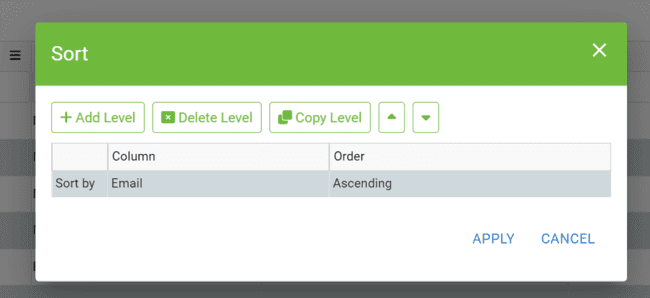
Here hidden value is email**.** By unchecking the tick mark as above. User can hide upto one column left in the grid.

User cannot delete last column . if user try to hide the last column the error message pops up like this.



**Configure Sort**

Configure sort page is used to apply mutliple sorting conditions of different columns with add level,delete level, copy level, up & down for changes condition index level

****

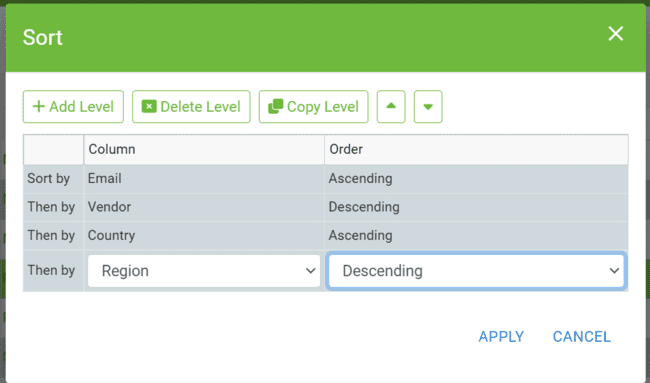
**Add level**

Add level can be used to apply multiple filters for column values in the grid.

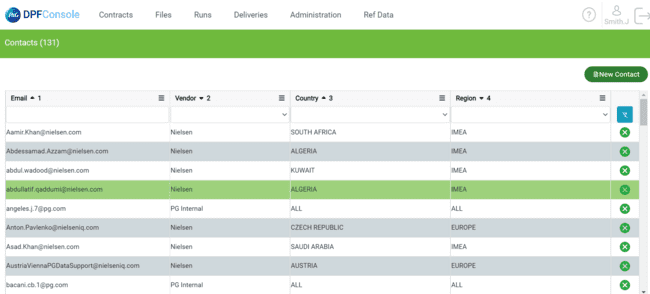
Graphical user interface, application, table

Description automatically generated

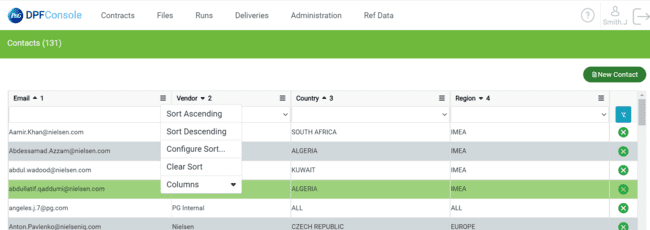
Here user can edit sort conditions by double click on the condition.



Click “Apply” for the changes. As above sort condition the numbering will be given in the grid

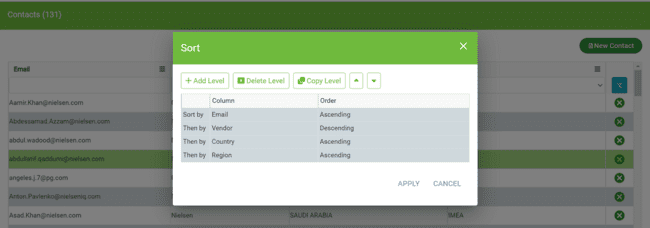


To remove configure sort. from multi drop down select “Clear Sort” .

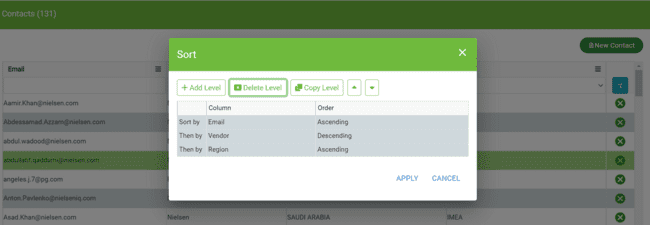


**Delete level**

This functionality deletes the selected sort condition in the configure sort. here is the initial stage of configure sort.



Example: user wants to delete a condition in configure sort i.e, country . Select country and click delete level.



**Copy level**

This functionality copies the same level condition user selected. Here is the initial page.

Graphical user interface, application, table

Description automatically generated

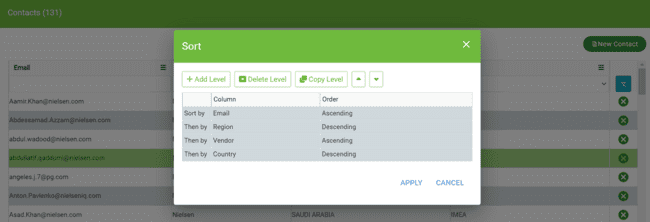
If user wants to copy level of vendor column condition, select vendor row and click on copy level.

Table

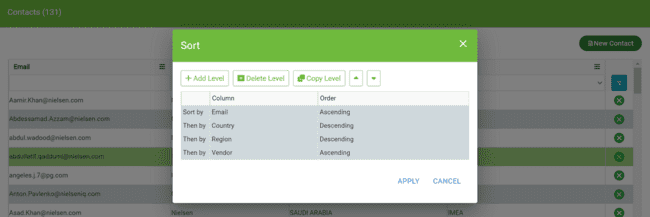
Description automatically generated

**Up arrow**

The up arrow changes the order of the selected condition towards top. Here is initial condition before using up arrow.



If user wants to change order of “country” next to “email”, select “country” then click up arrow two times.



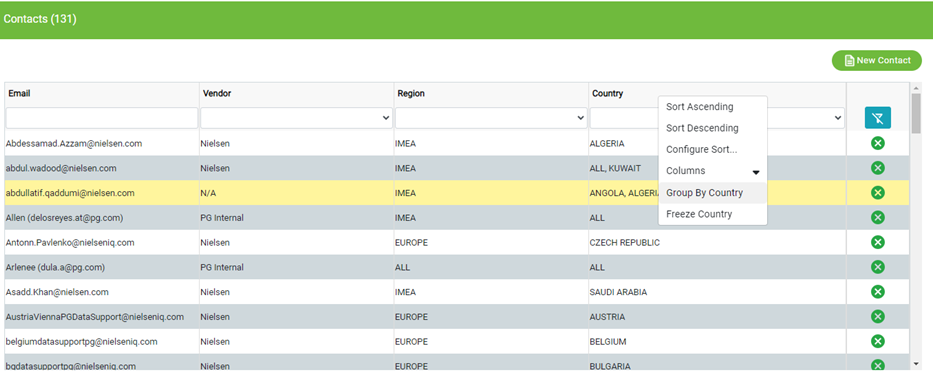
**Down arrow**

The Down arrow changes the order of the selected condition towards down/bottom. Here is initial condition before using down arrow.

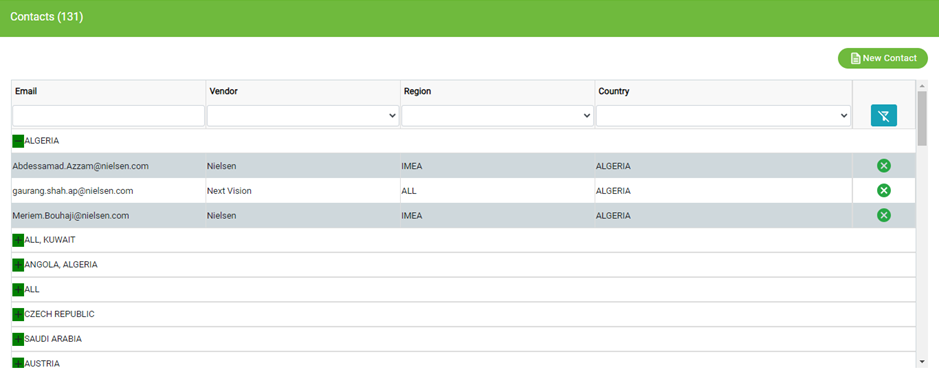
If user wants to change order of “Region” next to “Country”, select “Region” then click Down arrow two times.

Group By

Under column options there is a option Group By Column name which can be used to group table data having unique values as per group by column.

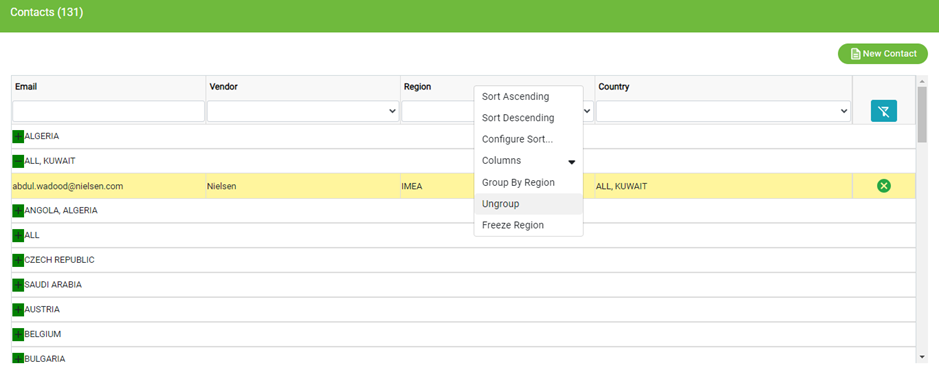


Click on group by column name, identical table rows grouped under parent node. If user want to expand and collapse, click on the parent node row. By default first parent node is expanded.

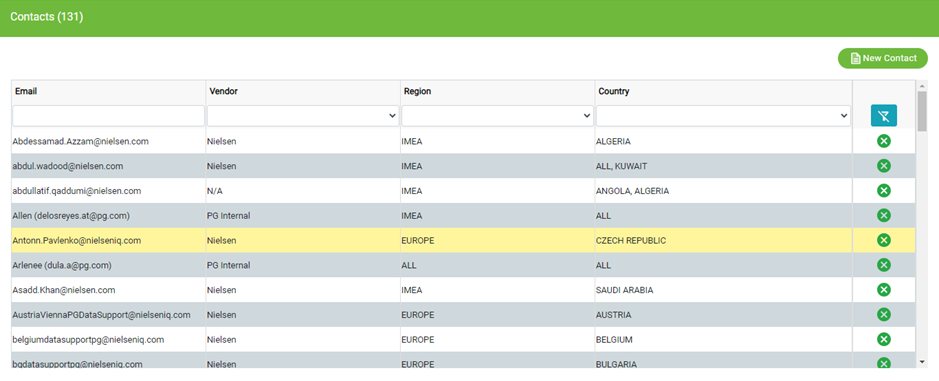


Ungroup

Under column options there is a option Ungroup which is only available when user apply group by to any column.

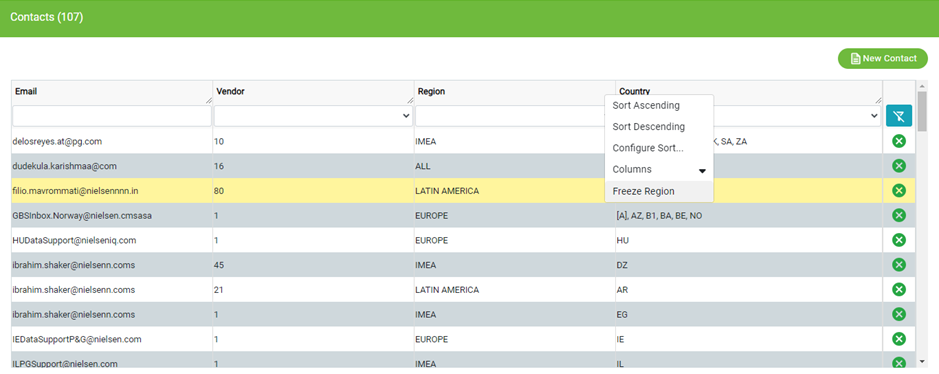


It is used to ungroup table data which is group by any column.

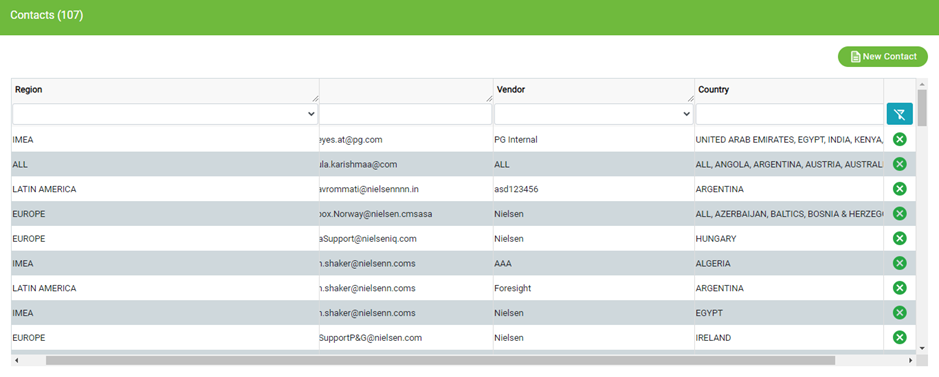


Freeze

Under column options there is a option Freeze Column name which is used to freeze column.

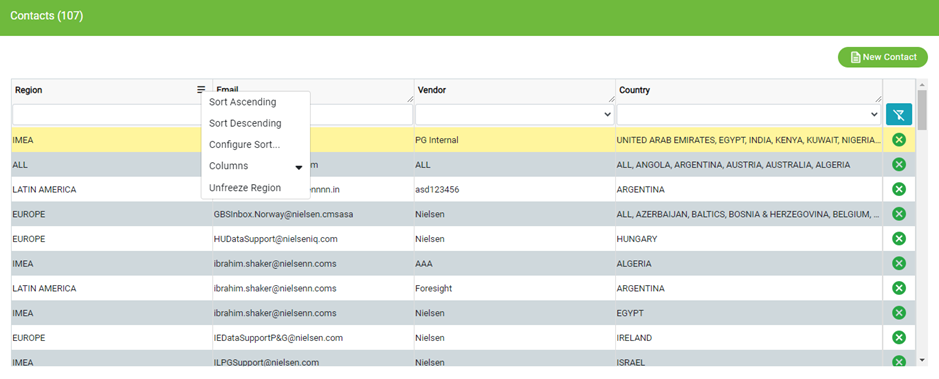


When user freeze any column, it fixed that column on first position in the table and it can’t be horizontally scrollable.

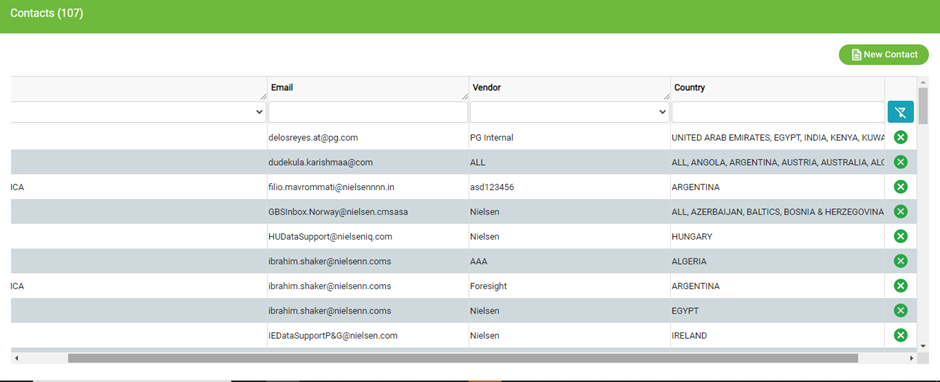


Unfreeze

Under column options there is a option Unfreeze Column name which is only available for that columns which are freezed.



It is used to unfreeze column which are freezed, then user can scroll that unfreeze columns horizontally.



**API calls Contacts**

|  |  |  |
| --- | --- | --- |
| **Put /Contacts/modifyContact** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/modifyContact** | **Api modifies contacts data- (payload not required)** |
| **Post**  **/Contacts/AddContact** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/addContact** | **Api adds contact data- (payload not required)** |
| **Get**  **/Contacts/search** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/search?limit=75&page=1** | **Api for search-(payload not required)** |
| **Get /Contacts/readVendors** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/readVendors** | **Api for readingvendor data-**  **(Payload not required)** |
| **Get /Contacts/readRegions** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/readRegions** | **Api for readregions- (payload not required)** |
| **Get /Contacts/readCountries** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/readCountries** | **API for search- (payload not required)** |
| **Get /Contacts/readContacts** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/readContacts?startRow=0&endRow=75** | **API to readcontacts- (payload not required)** |
| **Delete /Contacts/deletecontact** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/deleteContact?id=**  **101%25231%2523IMEA** | **API for deleting contact data- payload required.**  **Input=**  **101%25231%2523IMEA** |

## Measures

To interact with the “Measures” page. From Menu choose “Ref Data” multi drop down choose “Measures”.

Measures page deals with vendor submitted details- Info page Name, Business Name, DB Name, Product(aggregate), Area(aggregate), time(aggregate) of product data.

A screenshot of a computer

Description automatically generated

The below image represents “Measures”.

1. Fact Type, Product, Area, Time have Multi dropdown.
2. DB Name, Business Name, info Page Name have search option.

**Features:**

Refresh button

Functionality is to refresh grid to allow new entities of data in the grid. Button is located at the “top-right” just above the “funnel”.

A screenshot of a computer

Description automatically generated

Edit Measure

To edit a measure detail of user, click “edit”. This directs to the below image, here user can edit the aggregation method and description only. And able to add a “New Tag” for the data.

Corresponding to it.

Graphical user interface

Description automatically generated

Business Name, Fact Type, Name in DB, Formula are the default one user cannot edit these details.

New Tag

New tag allows user to add the vendor, Tag, Country, Contract Code for measure using multi drop down, while selecting Tag details user can choose only one option from country, Contract Code. click “Enter” to save or use “Save button” below.

A screenshot of a computer

Description automatically generated

After Save, message pops as below for confirmation to update the measure.

A screenshot of a phone

Description automatically generated

After Save changes it will give us confirmation as below.

A screenshot of a computer

Description automatically generated

API Calls – Measures:

|  |  |  |
| --- | --- | --- |
| Put /Measures/modifyMeasures | http://cloudpanel-ui-api.azurewebsites.net/dpf/ measures/modifyMeasure | API calls for modification of measures data.  (No payload)) |
| **Put**  /Measures/editMeasureTag | http://cloudpanel-ui-api.azurewebsites.net/dpf/ measures/editMeasureTag | API calls for editingmeasuretag of measures data.  (No payload)) |
| **Post**  /Measures/addMeasureTag | http://cloudpanel-ui-api.azurewebsites.net/dpf/measures/addMeasureTag | API calls for addmeasuretag of measures data.  (No payload)) |
| **Get /Measures/search** | http://cloudpanel-ui-api.azurewebsites.net/dpf/measures/search?limit=75&page=1 | Api calls for searching measures data.  (No payload)) |
| **Get /Measures/searchTag** | http://cloudpanel-ui-api.azurewebsites.net/dpf/measures/searchTag?limit=75&page=1 | Api calls for searchingtag of measures data.  (No payload)) |
| **Get /Measures/readMeasure** | http://cloudpanel-ui-api.azurewebsites.net/dpf/measures/readMeasureTags?measureId=1 | Api calls to readmeasures of measures data.  (No payload)) |
| **Get /Measures/getMeasureList** | http://cloudpanel-ui-api.azurewebsites.net/dpf/measures/getMeasureList?startRow=0&endRow=75 | Api calls to getmeasurelist details of measures data.  (No payload)) |
| **Delete /Measures/deleteMeasureTag** | http://cloudpanel-ui-api.azurewebsites.net/dpf/measures/deleteMeasureTag?  axlryRcdListId=32333 | Api calls deleteMeasuretag details of measures data.  (Payload required  axlryRcdListId=  32333  ) |

API Calls- Measures

## Products

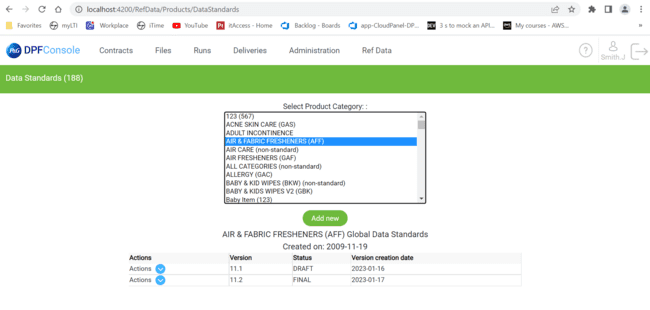
### Data Standards

Data standards screen is to add a category, edit the existing category, to perform some actions like preview, download the details of the product.



Data Standards deals with the product data, the way each product is categorized, description, comments, and the attachments of the product.

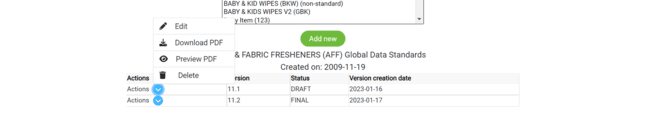
By selecting a product in Product Category users can see several details as below.



Here in example AIR &FABRIC FRESHENERS is Category Name, AFF is Category attribute Value (3-letter code). Created date of the product details, version, Status: Draft or Final, version creation date, actions able to perform on the product will be provided.

**Features:**

**Actions:**

****

From dropdown users can edit the product details, Download PDF, Preview PDF, Delete the Version of “draft” only.

**Download PDF**

PDF will be downloaded in zip format only.



**Preview PDF**

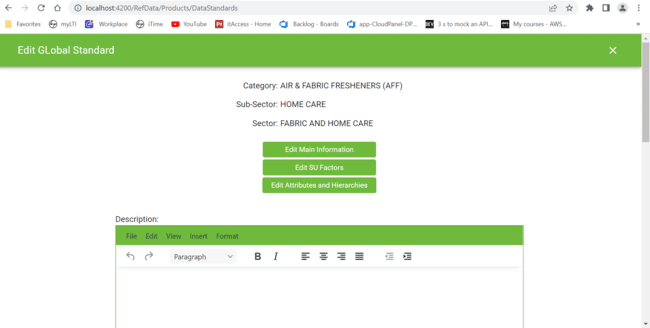
This functionality will redirect you to the default browser where you can see product details.

Graphical user interface, text

Description automatically generated

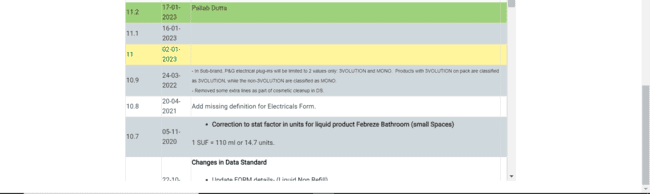
**Edit**

Edit button redirects user to “Edit Global Standard” like below Image.



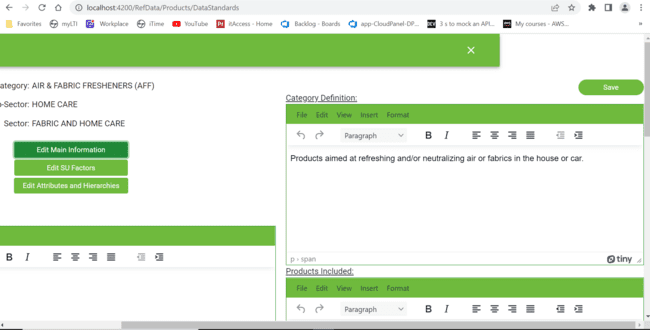
**Graphical user interface, text, application

Description automatically generated**

****

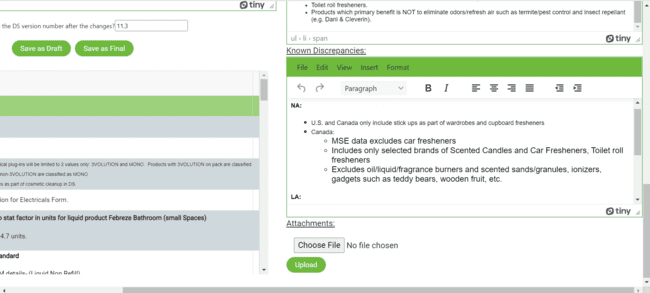
Here description and comment of user will be added in descriptive way. In tinymce user can take a fresh doc, Edit the previous description, view, insert a horizontal line, formatting the written doc.

**Edit Main Information**

****

**Graphical user interface, text, application

Description automatically generated**

****

Users can edit the main information of a product, save the changes for confirmation. upload a file about a product like PDF, Zip file, JPG, JPEG. confirmation message pops like.

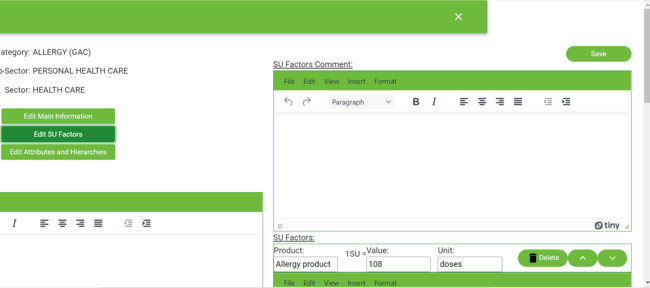


Save as Draft with same version name will append the newly edited data to the previous draft.

If the user wants to view a separate draft change the version name. for committing the changes opt the Final.

**Edit SU Factors**

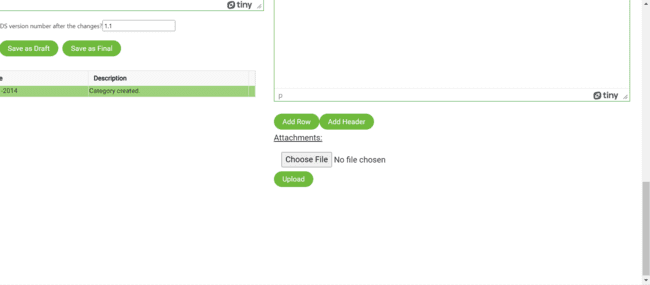
**Image 1**

****

**Image 2**

**Graphical user interface, application, table

Description automatically generated**

****

Add row- creates a new tinymce like “image 2”. To compare product with value, unit.

Add Header -creates a paragraph tinymce as “image 1”. To add description.

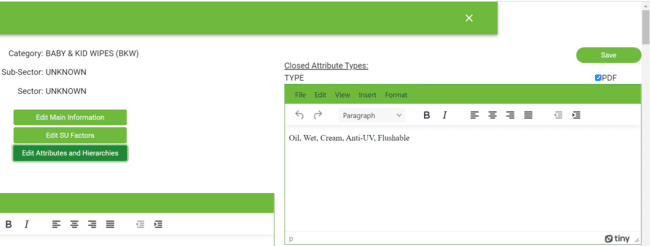
**Edit Attributes and Hierarchies**

Attributes falls under 2 categories: closed type attributes, open type attributes.

**Closed attribute types.**

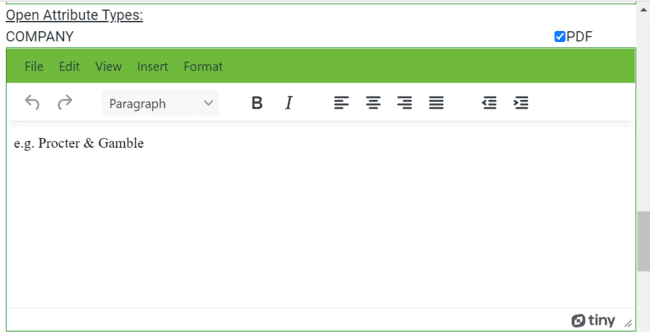
1. Type
2. Thickness
3. Segment
4. Tier
5. Brand etc.

Each Attribute falls in separate tinymce. Checkbox is enabled to show the data of attribute in pdf otherwise disabled the checkbox.

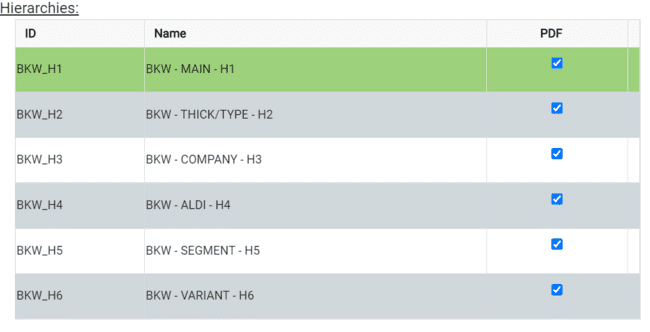


**Open attribute types**

1. Company
2. Variant
3. Item
4. Pack type
5. Scent
6. Category etc.

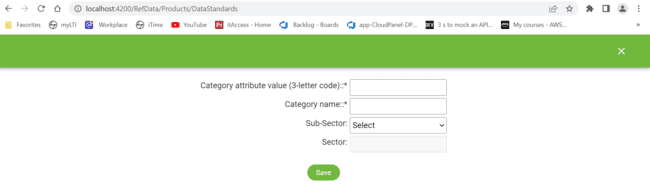
****

**Hierarchies**

****

**Add new.**

Users can add new product details using “Add new.”

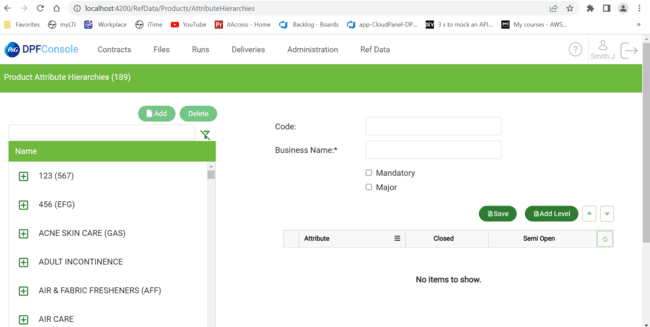


|  |  |  |
| --- | --- | --- |
| **Post**  **/dataStandard**  **/uploadAttachment** | <http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/uploadAttachment>?  dataStandardId=280&sectionCode=KNDISC | **Api calls for uploadattachment.**  **(payload required-datastandardid,sectioncode)** |
| **Post**  **/dataStandard**  **/saveDataStandard** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/saveDataStandard | **Api calls for saving datastandard**  **(Payload not required)** |
| **Post**  **/dataStandard**  **/saveCategory** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/saveCategory | **Api calls for saving category.**  **(Payload not required)** |
| **Get**  **/dataStandard**  **/versionHistory** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/versionHistory?categoryId=GAF | **Api calls for version history**  **(payload required-datastandardid, categoryid\*)** |
| **Get**  **/dataStandard**  **/previewPdf** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/previewPdf?dataStandId=280 | **Api calls for preview pdf**  **(payload required-datastandardid\*)** |
| **Get**  **/dataStandard**  **/hierarchies** | <http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/hierarchies>?  dataStandId=280&categoryId=GAF | **Api calls for hierarchies**  **(payload required-datastandardid\*, categoryid\*)** |
| **Get**  **/dataStandard**  **/getSubGBU** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/getSubGBU | **Api calls for getsubgbu**  **(payload not required)** |
| **Get**  **/dataStandard**  **/getStatuses** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/getStatuses | **Api calls for getstatuses**  **(payload not required)** |
| **Get**  **/dataStandard**  **/getGBU** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/getGBU | **Api calls for getgbu**  **(payload not required)** |
| **Get**  **/dataStandard**  **/getDataStandardForList** | [http://cloudpanel-ui- api.azurewebsites.net/dpf/dataStandard/](http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/)getDataStandardForList?categId=280 | **Api calls for getDataStandardForList**  **(payload required**  **Categid\*)** |
| **Get**  **/dataStandard**  **/getDataStandardDetails** | <http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/getDataStandardDetails>  ?dataStandId=280&categoryId=GAF | **Api calls for getDataStandardDetails (payload required-datastandardid\*, categoryid\*)** |
| **Get**  **/dataStandard**  **/getCategories** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/getCategories | **Api calls for getCategories**  **(payload not required)** |
| **Get**  **/dataStandard**  **/downloadAll** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/downloadAll?dataStandId=280 | **Api calls for downloadAll (payload required-datastandardid\*)** |
| **Get**  **/dataStandard**  **/ dataStandardExists** | <http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/dataStandardExists>?  categoryId=280&versionNum=10.3 | **Api calls for dataStandardExists (payload required-versionnum\*, categoryid\*)** |
| **Get**  **/dataStandard**  **/categoryExists** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/categoryExists?categoryId=gaf | **Api calls for categoryexists (payload required-categoryid\*)** |
| **Delete**  **/dataStandard**  **/deleteDataStandard** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/deleteDataStandard?dataStdId=280 | **Api calls for deleteDataStandard (payload required- datastdid\*)** |
| **Delete**  **/dataStandard**  **/deleteCategory** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/deleteCategory?categoryId=GAF | **Api calls for deletecategory (payload required-categoryid\*)** |
| **Delete**  **/dataStandard**  **/deleteAttachment** | http://cloudpanel-ui-api.azurewebsites.net/dpf/dataStandard/deleteAttachment?dataStdAttchId=12778 | **Api calls for deleteattachment (payload required- dataStdAttchId\*)** |

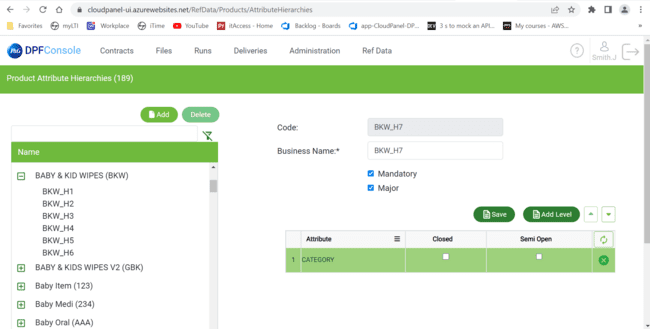
**API calls Data standards**

### Attribute Hierarchies

Attribute hierarchies has view of categories, code names of corresponding categories, vendor can select their product, to add levels with semi open or open to be viewed by P&G.



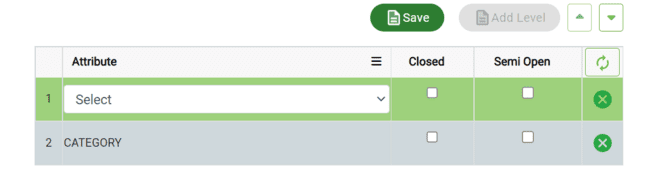
Attribute hierarchies initially “Add,” “Delete” buttons are disabled. The Add will enable the user select product name as below.



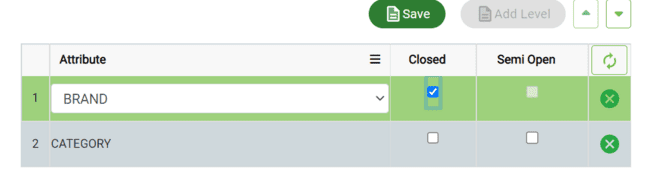
Here in image selected product “BABY & KID WIPES”. If user clicks Add, it adds new data

“BKW\_H7”.

User can “Add level” for new attributes (closed & open). Save for committing data successfully.



Select attribute and user can opt only one checkbox from “closed” and “semi open”. if user select one checkbox, another will be disabled.



“Save” for committing changes.



“delete” for removing attributes like “BKW\_H7” by selecting.



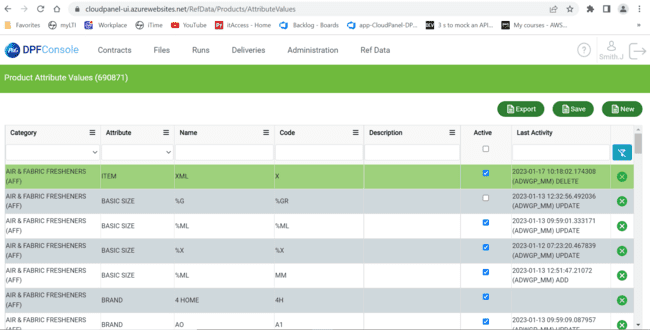
**API CALLS - Product Attribute Hierarchies**

|  |  |  |
| --- | --- | --- |
| Put/prodAttr/  modifyValue | <http://cloudpanel-ui->  api.azurewebsites.net/dpf/prodAttr/modifyValue | Api calls for modification of prodattr (payload not required) |
| Post/prodAttr/  addValue | <http://cloudpanel-ui->  api.azurewebsites.net/dpf/prodAttr/addValue | Api calls addvalue to prodattr (payload not required) |
| Get/ prodAttr/search | http://cloudpanel-ui-api.azurewebsites.net/dpf/prodAttr/search?limit=75&page=1 | Api calls for search  (payload not required) |
| Get/prodAttr/  Read | <http://cloudpanel-ui-api.azurewebsites.net/dpf/prodAttr/>  read?startRow=0&endRow=75 | Api calls to read  (payload not required) |
| Get/prodAttr/  readAttributes | <http://cloudpanel-ui->  api.azurewebsites.net/dpf/prodAttr/readAttributes | Api calls to readattributes  (payload not required) |
| Get/prodAttr/  proposeCode | <http://cloudpanel-ui-api.azurewebsites.net/dpf/prodAttr/proposeCode?categoryId=>  RII&prodAttrId=45&prodAttrValName=CATEGORY | Api calls to propose code  (payload required-  Categoryid\*,prodattrid\*, prodattrvalname\*) |
| Delete/prodAttr/  deleteValue | <http://cloudpanel-ui-api.azurewebsites.net/dpf/prodAttr/>  deleteValue?prodAttrValId=45 | Api calls to deletevalue  (payload required- prodattrvalid\*) |

### Attribute Values

Product attribute values have product category (same as data standards) with attribute values.

This screen is used to export the complete details of the categories with last activity.



**Features:**

Edit

Users can edit Name, Description, Last Activity by “double click” on row entity.



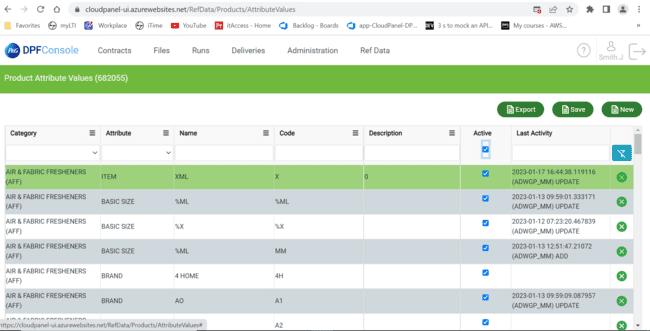
Export

Exports the total data in excel format.



Active

Active checkbox filters out the active attribute values which are in data (frequently used).



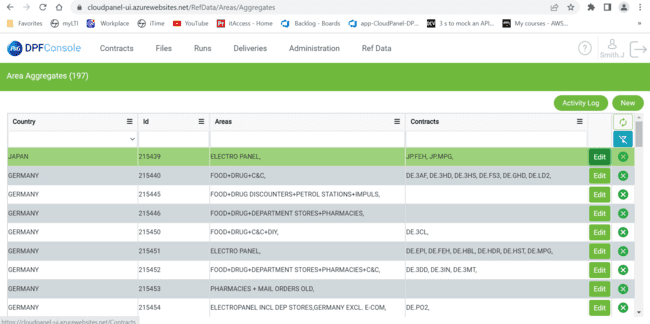
**API Calls -attribute values**

|  |  |  |
| --- | --- | --- |
| Get/prodAttrValueExport/  readCategoryAndAttributeById | <http://cloudpanel-ui-api.azurewebsites.net/dpf/>  prodAttrValueExport/readCategoryAndAttributeById | Api calls to readCategoryAndAttributeById (payload not required) |

## Areas

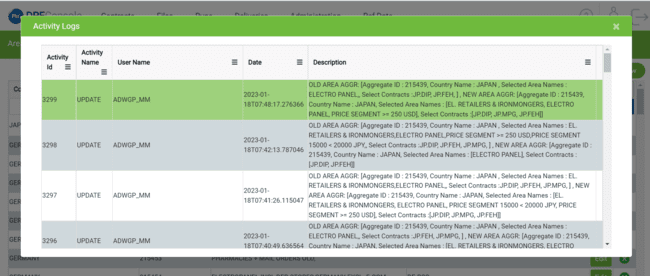
### Area Aggregates

Area aggregates screen is to categorize areas and contracts for a selected id with country. Here P&G can select areas and contracts from the available area’s vendor provided.



Area aggregates deals with data of a specific country with corresponding “ID” of it. It consists of Aggregated Areas, Assigned Contracts details of country categorized as “Available”, “Selected”.

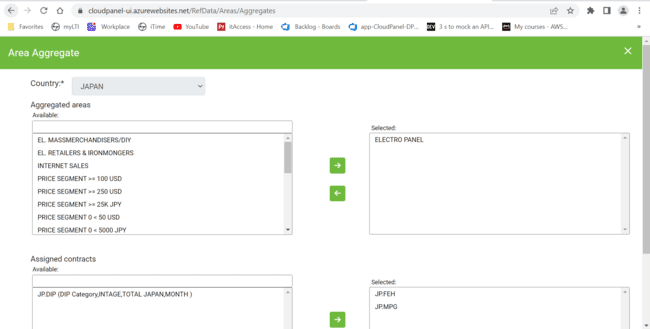
**Activity log**

****

Logs define the changes that user made to the data, With activity id, Activity Name, Username, Description about log.

**Edit**

User can choose available Aggregated Areas, Assigned Contracts by “edit” option.



Details are arranged by selecting Areas or Contracts, shift them by using Arrow symbols from Available to Selected vice versa. Save by using the save button.



Example:

**Before**



**After**

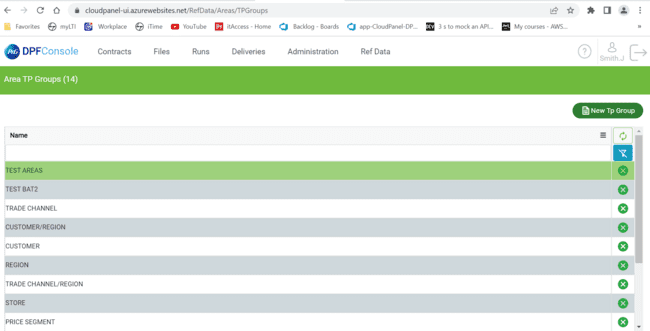


**API Calls Area Aggregates**

|  |  |  |
| --- | --- | --- |
| Put/areaAggregates  /Update | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/update | Api calls to update  (payload not required) |
| Post/ areaAggregates  /addAggregates | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/addAggregates | Api calls to addaggregates  (payload not required) |
| Get/ areaAggregates/  selectedContract | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/selectedContract?aggregateId=215439 | Api calls for selectedcontract  (payload required-aggregateid\*) |
| Get/ areaAggregates/  selectedAreas | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/selectedAreas?aggregateId=10 | Api calls for selectedareas  (payload required-aggregateid\*) |
| Get/ areaAggregates/  search | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/search?limit=75&page=1 | Api calls to search  (payload not required) |
| Get/ areaAggregates/  readAreaGroups | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/readAreaGroups?startRow=0&endRow=75 | Api calls to readareagroups  (payload not required) |
| Get/ areaAggregates/  fetchVendorName | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/fetchVendorName?vendorId=782195 | Api calls for fetchvendorname  (payload required-vendorid\*) |
| Get/ areaAggregates/  availContract/Search | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/availcontract/search?limit=75&page=1 | Api calls to availcontract/search  (payload not required) |
| Get/ areaAggregates/  availableContract | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/availableContract?aggregateId=76 | Api calls for availablecontract  (payload required-aggregateid\*) |
| Get/ areaAggregates/  availableAreas | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/availableAreas?aggregateId=47 | Api calls for availableareas  (payload required-aggregateid\*) |
| Get/ areaAggregates/  allContracts | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/allContracts?countryId=215439 | Api calls for allcontracts  (payload required-countryid\*) |
| Get/ areaAggregates/  allAreas | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/allAreas?country=JAPAN | Api calls for allareas  (payload required - country\*) |
| Get/ areaAggregates/  addContract/search | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/addContract/search?limit=75&page=1 | Api calls toaddcontract/ search  (payload not required) |
| Get/ areaAggregates/  addArea/search | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/addArea/search?limit=75&page=1 | Api calls to addarea/search  (payload not required) |
| Delete/  areaAggregates  /delete | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/delete?id=215439 | Api calls for delete  (payload required id\*) |
| Get  /areaAggregates  /availArea/search | http://cloudpanel-ui-api.azurewebsites.net/dpf/areaAggregates/availarea/search?limit=75&page=1 | Api calls to availarea/search  (payload not required) |

### TP Groups

TP groups are used to add new group or region to the product.



**Features**:

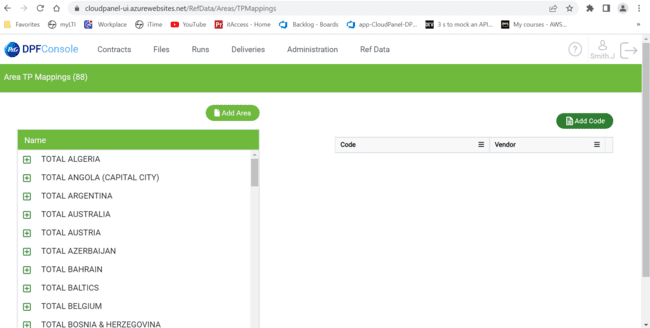
* Edit by “double click”.
* Delete by using “X”.
* Add by using “New TP Group”.
* Save data by using “enter”.
* Refresh
* Funnel filter

**API Calls TP Groups**

|  |  |  |
| --- | --- | --- |
| **PUT/area/**  **modifyAreagroup** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/area/modifyAreaGroup** | **Api call for modifyarea group**  **(payload not required)** |
| **Post/area/**  **addAreagroups** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/area/addAreaGroups** | **Api call for addareagroup group**  **(payload not required)** |
| **Get/area/search** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/area/search?limit=75&page=1** | **Api call for search group**  **(payload not required)** |
| **Get/area/**  **getAreaGroups** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/area/getAreaGroups?startRow=0&endRow=75** | **Api call for getarea groups**  **(payload not required)** |
| **Delete/area/**  **deleteAreaGroup** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/area/deleteAreaGroup?mktGrpId=10** | **Api call to deletearea group**  **(payload required**  **-mktgrpid\*)** |

### TP Mappings

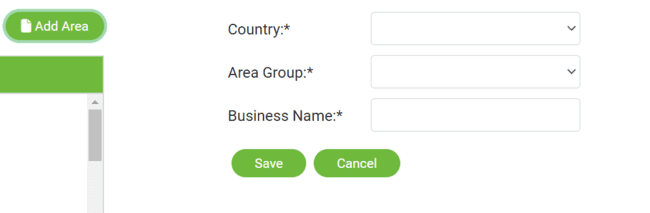
This screen will interact with vendors to add code or edit, to a group in a specific country.



Features:

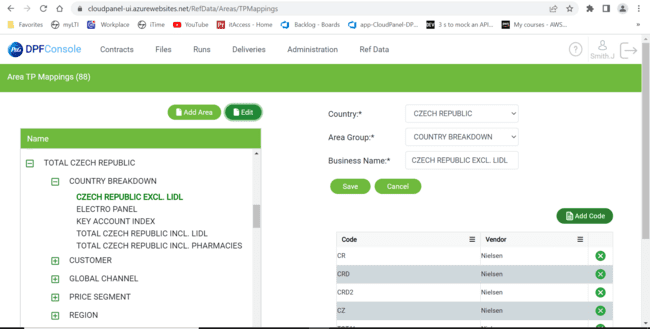
Add Area

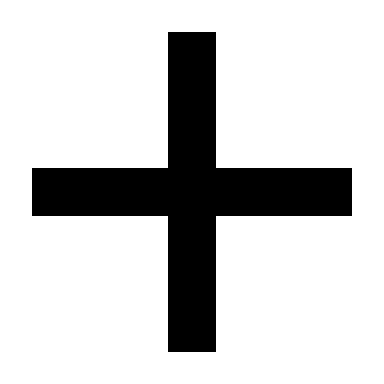
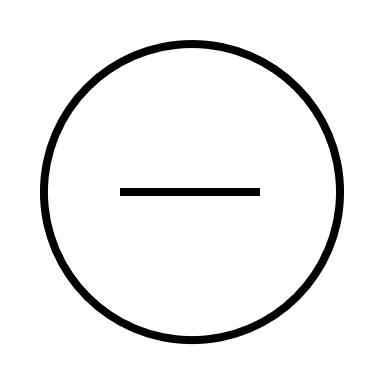
Add area functionality is to entry data of country, Area Group, Business Name (should be unique). Mandatory to fill.



**Edit**

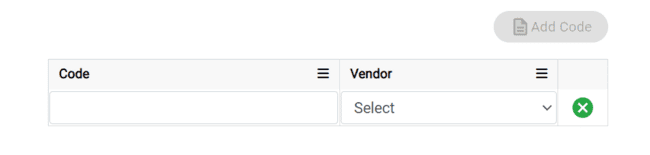
Edit option comes into picture when user selects “Business Name”.



By elaborating  , first one is “country”, next “Area Group” last one “Business Name”. to hide details, use  symbol.

**Add code.**

Add code be used to add code for vendor at any stage (country, area group, Business Name).



**Save**

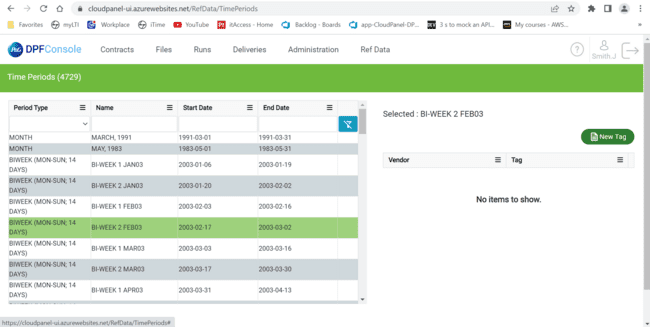
User can save data by using save button or by using enter.

**API Calls- TP Mappings**

|  |  |  |
| --- | --- | --- |
| **Put/tpMapping/**  **modifyArea** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/modifyArea** | **Api calls to modify area**  **(payload not required)** |
| **Post/tpMapping/**  **addcode** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/addCode** | **Api calls to addcode**  **(payload not required)** |
| **Post/tpMpapping/**  **addArea** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/addArea** | **Api calls to add area**  **(payload not required)** |
| **Get/tpMapping /**  **readNodes** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/readNodes?parentData=null** | **Api calls to readnodes**  **(payload not required)** |
| **Get/tpMapping/**  **readCodes** | [**http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping**](http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping)  **/readCodes?startRow=0&endRow=75&areaCode=206772** | **Api calls to readcodes**  **(payload required**  **-areacode\*)** |
| **Get/tpMapping/**  **readCodes/**  **search** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/readCodes/search?limit=75&page=1** | **Api calls to modify area**  **(payload not required)** |
| **Get/tpMapping/**  **getParquetFile** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/getParquetFile** | **Api calls to modify area**  **(payload not required)** |
| **Get/tpMapping/**  **getArea** | [**http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/**](http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/)  **getArea?startRow=0&endRow=75&areaCode=206772** | **Api calls to getarea**  **(payload required**  **-areacode\*)** |
| **Delete/tpMapping/**  **deleteCode** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/tpMapping/deleteCode?id=211084** | **Api calls to delete code**  **(payload required**  **-id\*)** |

## Time periods

To filter the records of products or to search within a period. Vendors can add a new tag for a selected period.



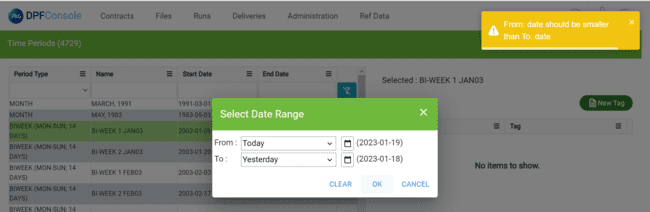
**Features:**

**Start Date & End Date**

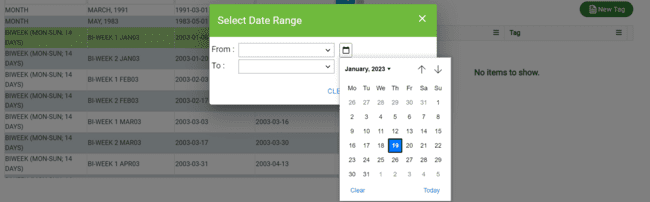
Start Date &end Date of data can be chosen by using calendar (specific dates) or multidrop down (string data like today, yesterday) as below.



Note: From: date should be samller than To: date or “same day”. In both “start Date” and “End Date”.

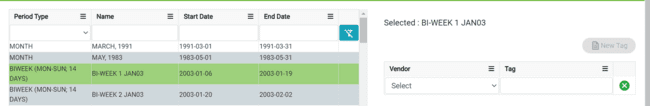


User can choose date by using calender also.



**New Tag**

New tag will add vendor with a tag for selected “TimePeriod”.

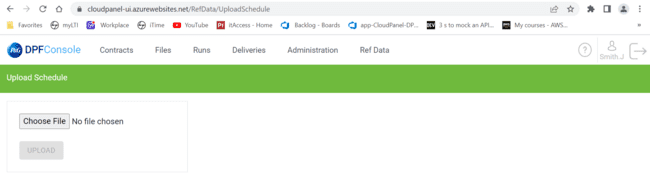


**API Calls – Time Period**

|  |  |  |
| --- | --- | --- |
| **Put/timePeriod/modifyTag** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/modifyTag** | **Api call to modify tag**  **(payload not required)** |
| **Post/timePeriod/addTag** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/addTag** | **Api call to addtag**  **(payload not required)** |
| **Get/timePeriod/tags/search** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/tags/search?limit=75&page=1** | **Api call to search a tag**  **(payload not required)** |
| **Get/timePeriod/search** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/search?limit=75&page=1** | **Api call to search**  **(payload not required)** |
| **Get/timePeriod/read** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/read?startRow=0&endRow=75** | **Api call to read**  **(payload not required)** |
| **Get/timePeriod/getTagList** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/getTagList?timePeriodId=1021** | **Api call to gettaglist**  **(payload required**  **-timeperiodid\*)** |
| **Delete/timePeriod/deleteTag** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/timePeriod/deleteTag?axlryRcdId=7727** | **Api call to deletetag**  **(payload required**  **-axlryrcdid\*)** |

## Upload Schedule

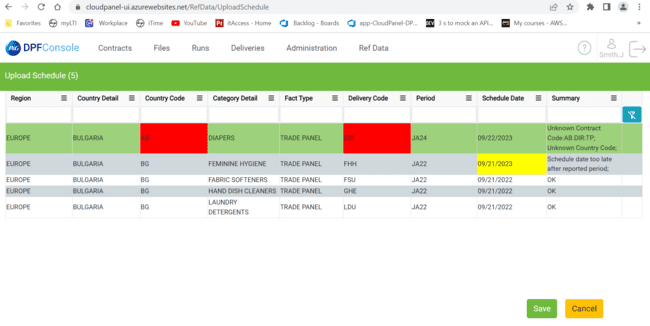
upload schedule is to check the errors in excel sheet before saving to the backend.



“Choose file” should be excel format for upload that too in required format to fit in the grid. It does not support any other format.

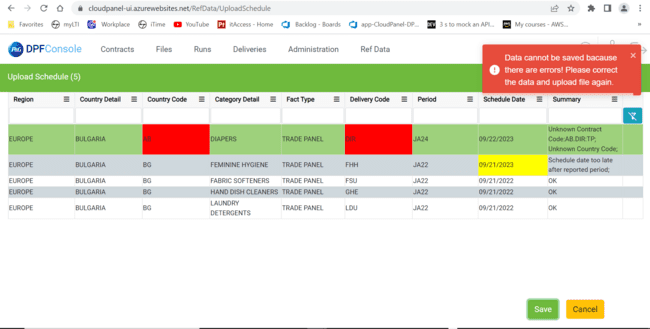


Upload button enables after user choose file. Upload for grid format of data.

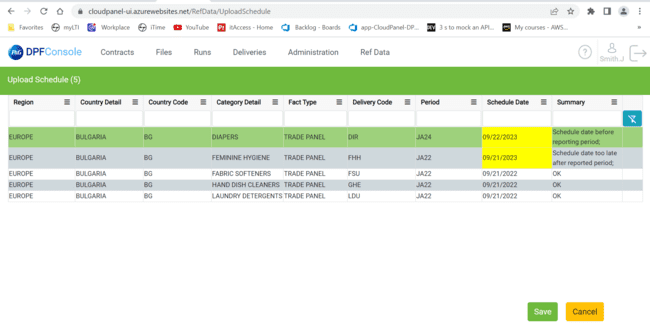


After uploading a file “reds” are errors. “**yellows** ” are warnings. user need to remove all errors in excel sheet to save in backend. Summary should be “ok” or with warning.

if user try to upload without changing errors, it thrioughs error as below.



Correct the data using “summary” and upload the file again as below and Save it.



Now data sucessfully saved.

**API Calls Upload Schedule**

|  |  |  |
| --- | --- | --- |
| **Post/uploadSchedule**  **/uploadExcel** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/uploadSchedule/uploadExcel** | **Api call to uploadexcel**  **()payload not required)** |
| **Get/uploadschedule**  **/Search** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/uploadSchedule/search?limit=75&page=1** | **Api call to search**  **()payload not required)** |
| **Get/uploadschedule**  **/saveData** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/uploadSchedule/saveData** | **Api call to savedata**  **()payload not required)** |
| **Get/uploadschedule**  **/getResults** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/uploadSchedule/getResults?startRow=0&endRow=75** | **Api call to getresults**  **()payload not required)** |
| **Get/uploadschedule**  **/getErrorDescriptions** | **http://cloudpanel-ui-api.azurewebsites.net/dpf/uploadSchedule/getErrorDescriptions** | **Api call to geterrordescriptions**  **()payload not required)** |

## HHP Mappings:

**NOTE – HHP mapping DD api subscription expires in 1 year and that must be renewed in 10 months.**

HHP Mappings screen contains the ability to manage below target datasets.

1. MM\_HHP\_BUYR\_GRP\_CLASS
2. MM\_HHP\_TIME\_MAPNG
3. MM\_HHP\_GEO\_CODE\_MAPNG
4. MM\_HHP\_ONE,
5. MM\_HHP\_TWO,
6. MM\_HHP\_THREE
7. MM\_HHP\_DATA\_CNTXT\_MAPNG

**Menu:**

To access the HHP Mappings in menu “Ref Data” Drop down, select HHP Mappings.

A screenshot of a computer

Description automatically generated with medium confidence

### MM\_HHP\_DATA\_CNTXT\_MAPNG

**Menu:**

To access the contacts in menu “Ref Data” Drop down, select MM\_HHP\_DATA\_CNTXT\_MAPNG under HHP mappings.

A screenshot of a computer

Description automatically generated with medium confidence

**Title:**

Title of the page “TARGET DATA SET: MM\_HHP\_DATA\_CNTXT\_MAPNG” with the count of entities in () bracket.

TARGET DATA SET: MM\_HHP\_DATA\_CNTXT\_MAPNG (43)

**Grid:**

Grid with column values source type and source details.

A screenshot of a computer

Description automatically generated with medium confidence

**Features:**

Import

Import is a feature enabled by clicking on the import button.

A screenshot of a computer

Description automatically generated with medium confidence

Edit

Edit is a feature enabled by double clicking on the fields like source type and source details to change the details or modifying purpose. Changes apply after “enter” button.

A screenshot of a computer

Description automatically generated with medium confidence

Delete Contact

Contacts can be deleted by selecting the checkbox and clicking ‘delete selected’ option.

A screenshot of a computer

Description automatically generated

If user tries to delete the data then confirmation message popups as per below picture.

A screenshot of a computer

Description automatically generated with medium confidence

Add Row

This page enables users to add a contact using “Add Row” button. That creates a row in Grid. Users can put details in the newly added row. To Save User need to press the “Enter” Button.

mm\_mktA screenshot of a computer

Description automatically generated with medium confidence

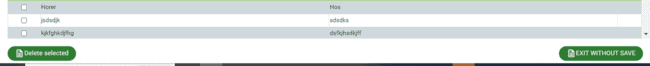
Export

Exports the total data in excel format.



Exit without save.

Users want to come out from the update or add row, they can click on exit without save.



**API calls – HHP Mappings**

|  |  |  |
| --- | --- | --- |
| **Post**  **hhpdatasetsmappings/saveDataSetsMapping** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/hhpdatasetsmappings/saveDataSetsMapping** | **Api adds/modifies data- (payload required fileType: HHP\_DATA\_CNTXT)** |
| **Get**  **/hhpdatasetsmappings /search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/hhpdatasetsmappings/search?page=1&limit=75&fileType=HHP\_DATA\_CNTXT&sort=+sourceType,+sourceDetails** | **Api for search (payload required fileType: HHP\_DATA\_CNTXT)** |
| **Delete / hhpdatasetsmappings /deletecontact** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/hhpdatasetsmappings/deleteDataSetMappings** | **API for deleting contact data- payload required fileType: HHP\_DATA\_CNTXT,**  **dataSetsMappingIds:[46, 8]** |
| **Import / hhpdatasetsmappings/uploadExcelFile** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/hhpdatasetsmappings/uploadExcelFile?fileType=HHP\_DATA\_CNTXT** | **Api for import (payload required fileType: HHP\_DATA\_CNTXT)** |
| **Export / hhpdatasetsmappings/ hhpDataSetExport** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/hhpdatasetsmappings/hhpDataSetExport?fileType=HHP\_DATA\_CNTXT** | **Api for export (payload required fileType: HHP\_DATA\_CNTXT)** |

### MM\_HHP\_TIME\_MAPNG

The functionalities (Import, Export. Add row, delete, update, exit without save) are similar to MM\_HHP\_DATA\_CNTXT\_MAPNG (Refer 3.7.1)A screenshot of a computer

Description automatically generated with medium confidence

### MM\_HHP\_BUYR\_GRP\_CLASS

The functionalities (Import, Export. Add row, delete, update, exit without save) are similar to MM\_HHP\_DATA\_CNTXT\_MAPNG (Refer 3.7.1)

A screenshot of a computer

Description automatically generated with medium confidence

### MM\_HHP\_GEO\_CODE\_MAPNG

The functionalities (Import, Export. Add row, delete, update, exit without save) are similar to MM\_HHP\_DATA\_CNTXT\_MAPNG (Refer 3.7.1)

A screenshot of a computer

Description automatically generated with medium confidence

### MM\_HHP\_ONE

The functionalities (Import, Export. Add row, delete, update, exit without save) are similar to MM\_HHP\_DATA\_CNTXT\_MAPNG (Refer 3.7.1)

A screenshot of a computer

Description automatically generated with medium confidence

### MM\_HHP\_TWO

The functionalities (Import, Export. Add row, delete, update, exit without save) are similar to MM\_HHP\_DATA\_CNTXT\_MAPNG (Refer 3.7.1)

A screenshot of a computer

Description automatically generated with medium confidence

### MM\_HHP\_THREE

The functionalities (Import, Export. Add row, delete, update, exit without save) are similar to MM\_HHP\_DATA\_CNTXT\_MAPNG (Refer 3.7.1)

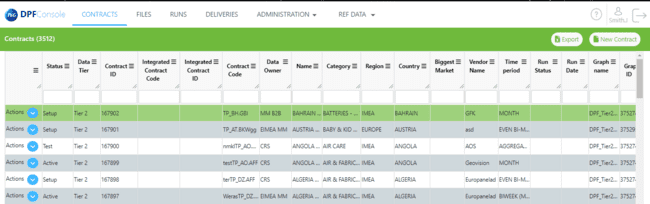
A screenshot of a computer

Description automatically generated with medium confidence

# Contracts

Contracts deals with Tier 1, Tier 2 and HHP data i.e., all type of data from trade panel (vendor collects data from traders of P&G product) or household panel (third party vendor collects information about P&G product directly from customers) and Sales fundamental.

Contracts in DPF console consists of multiple columns related to contracts dealt by P&G. All screens mentioned below are part of Contracts. Contracts UI is the main landing page in DPF console where we can create a new contract, edit a contract, and view all details of the contracts.

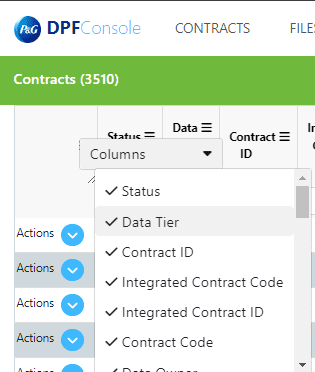


**These are the columns present in Contracts Screen respectively:**

* Status
* Data Tier
* Contract ID
* Integrated Contract ID
* Contract Code
* Integrated Contract Code
* Data Owner
* Name
* Category
* Region
* Country
* Biggest Market
* Vendor Name
* Time Period
* Run Status
* Run Date
* Graph Name
* Graph ID
* Promotion
* Data Auto Validation
* Master Validation Switch
* Ignore SU
* Currency
* Parallel Loading
* Physical Units
* Retention Period
* Source Details
* Source Type
* Source System
* Delivery Frequency
* Priority Flag
* Extract Refresh
* File Name From Vendor
* Onboarding ID
* Onboarding Name

# Columns Dropdown:

To add or remove the required columns click on hidden element dropdown in any of the columns. by using The down arrow symbol user can add, remove the columns.



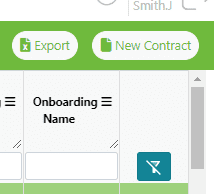
**Export**

If user wants to get all the details of a selected contract in contracts home page in an excel sheet format, then clicking on export will help to generate a excel with selected Contract Code in file name.



**Clear Filter**

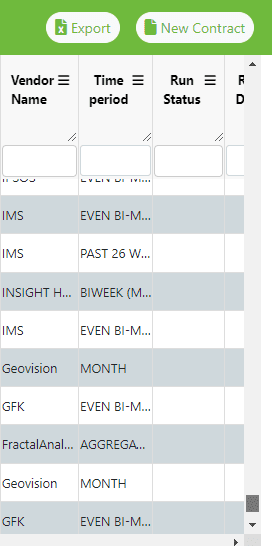
Filter is at top right located below the “New Contract.” Which removes all the Filters applied in the contracts page.



After clicking clear filter, all the default present data will be displayed in the UI.

**Infinite scroll**

It is a feature that helps with scrolling of grid until last entry of data.

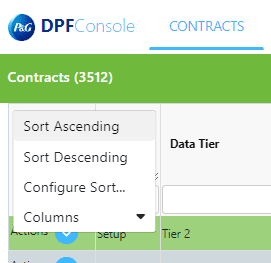


**Column options:**

By using this feature user can apply conditions for respective column as ascending, descending, configure sort, columns to be there in the grid,column options are breifly exlplained below.

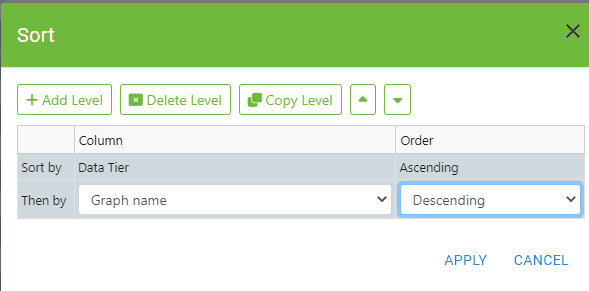
**Ascending & Descending**

Filtering out the values of user in the grid with ascending, descending order impacts the other columns value with same.



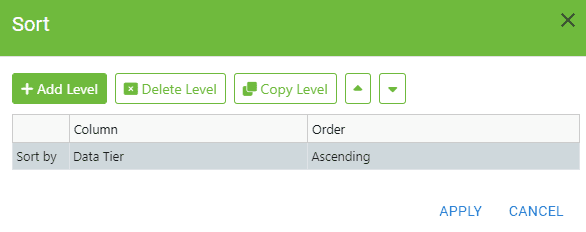
**Configure Sort**

Configure sort page is used to apply mutliple sorting conditions of different columns with add level,delete level, copy level, up & down for changes condition index level.



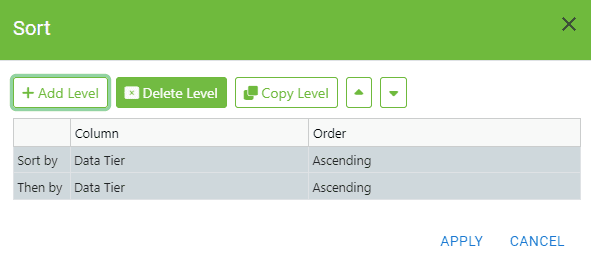
**Add level**

Add level can be used to apply multiple filters for column values in the grid.

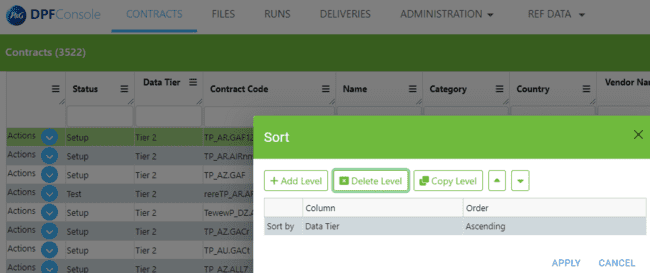


**Delete level**

This functionality deletes the selected sort condition in the configure sort. here is the initial stage of configure sort.

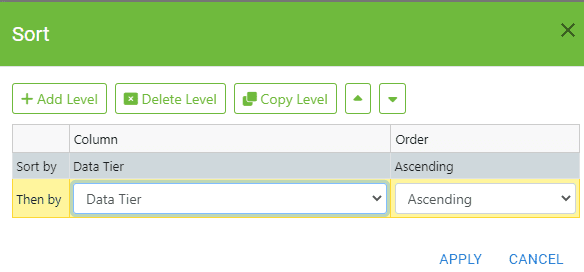


For Example, if the user wants to delete one record from above as there are duplicates, user can select that level and click on delete level then click Apply:



**Edit Custom Sort**

If user wants to change column name and ASC/DESC order then user can edit sort conditions by double clicking on the condition and Column name.

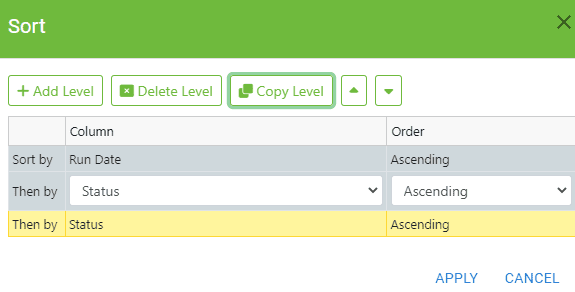


**Copy level**

This functionality copies the same level condition user selected. Here is the initial page.

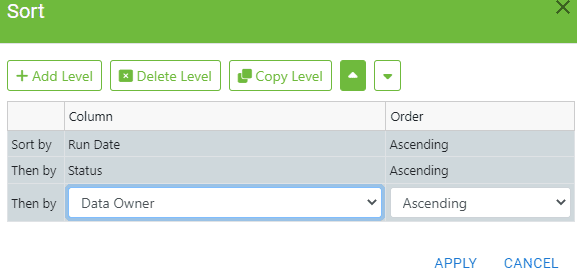


Example: If user wants to copy level of vendor column condition, select vendor row and click on copy level.

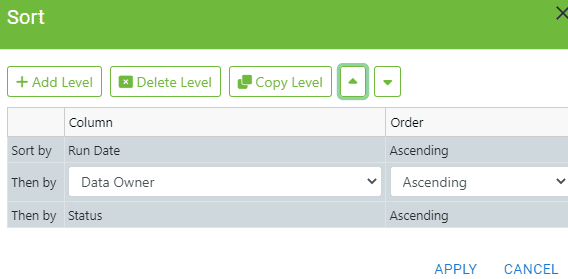


**Up arrow**

The up arrow changes the order of the selected condition towards top. Here is initial condition before using up arrow.

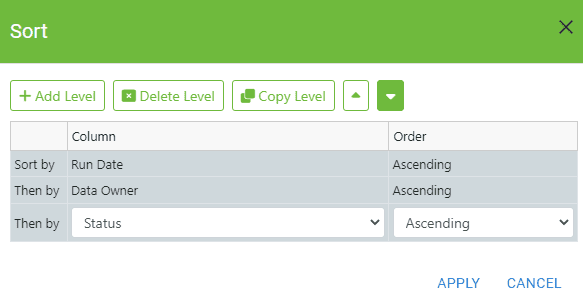


Example: If the user wants to Move the ‘Data Owner’row to move one level top then the up arrow must be clicked Once :

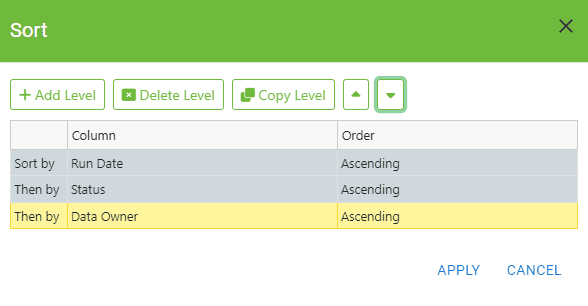


**Down arrow**

The Down arrow changes the order of the selected condition towards down/bottom. Here is initial condition before using down arrow.



Example if the user wants to take the ‘Data Owner Ascending’sort to one level down then user should click the down arrow



**New Contract:**

If the user wants to create a new contract then new contract option should be clicked and all the essential details should be filled which are required in contract from business perspective.



After Clicking the new contract button, another screen will open that has all the major details required to be filled in case of a contract creation.

**New Contract Creation :**

While creating a new contract a vendor can choose from three different Data tiers to create contract: Tier1, Tier2 and HHP.

And three different Fact type code’s namely: Trade Panel, Sales Fundamental and Household panel. And Type of environment the specific contract should be created should be selected mandatorily either Test or Production.

**Tier 2 Contract :**

To Create a tier2 contract, the vendor must choose tier2 data type and enter the Mandatory fields(\*) for the following below :

* + - * Geography
      * Currency
      * Category
      * Period Type
      * Retention Period
      * Vendor
      * Source system
      * Contract name
      * Contract Code
      * Data provider Key
      * File Name
      * File Pattern
      * Delivery Frequency
      * Processing Flow (Graph)
      * Data Owner

Based on Geography and category selected, the Contract Code gets automatically generated. And then Data Quality Validation page is appeared after saving the data selected in the New contracts Page.

# Tier 2 Contract Creation:

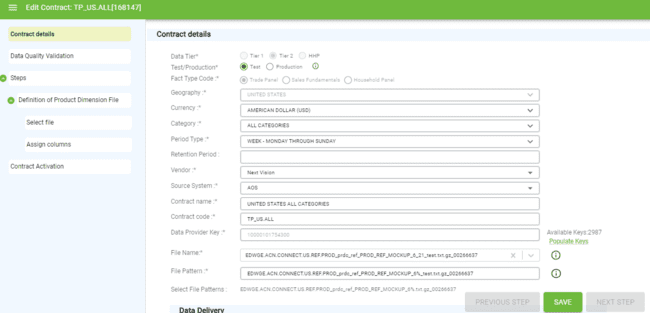
As we have two different categories for graphs like Single and Three file, we select Graphs according to that which map with respect to a single file Graph:

Below is the Graph for Three File contract creation:

* DPF\_Tier2\_Three\_Files\_Contract.

Below are the Graphs for Single File contract creation:

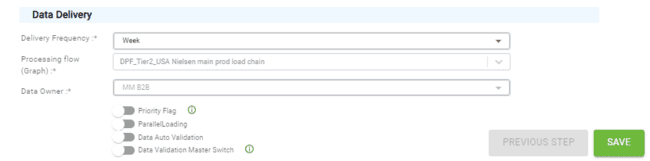
* DPF\_Tier2\_USA Nielsen fact item main chain
* DPF\_Tier2\_USA Nielsen fact categ main chain
* DPF\_Tier2\_USA Nielsen main prod load chain
* DPF\_Tier2\_Single\_Files\_Contract



Immediately after saving a contract, the page shows a pop-up that “Contract with ID XYZ has been Saved” and the contract will be now opened in EDIT CONTRACT mode ONLY.

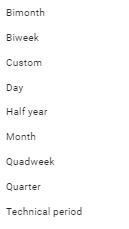
### Data Delivery:

In data delivery section we will be able to select The Delivery Frequency, Processing flow(graph), Data Owner of the contract that is getting created.

****

**Delivery frequency:**

This option enables user to select the delivery frequency time period of the contract which is getting created. The available Time period selection options are as follows :

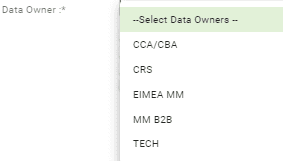


**Graph:**

Similar to Chain in legacy application, each graph contains various steps in the process flow/execution. Depending on Vendor file this graph has to be selected appropriately.

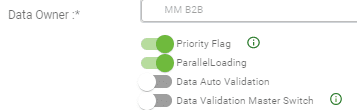
**Data Owner:**

This option enables user to select the type of business/ contract. The following options are available in dropdown to select:



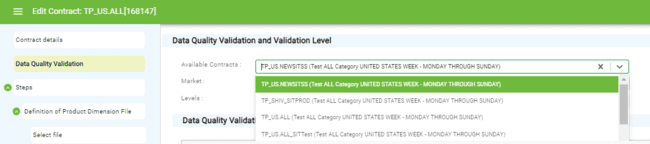
**Data Validation Master Switch:**

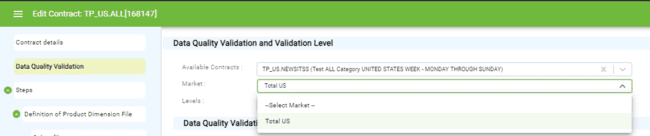
When this flag is enabled then only, we can select validation checks in checks screen.



### Data Quality Validation:

Here We can select the market option annd levels option to check the data quality validation and validation level table.





DQ Validation Table with the required values for populating the contract’s DQ : These are the respective Column Header’s :

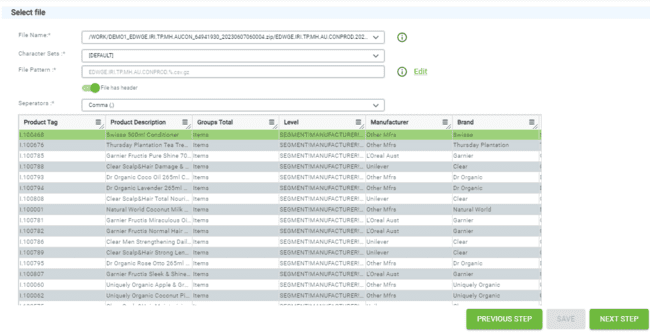
* ID
* Check Group
* Check Description
* Check Details
* Active (Y/N)



### Select File and Assign Columns:

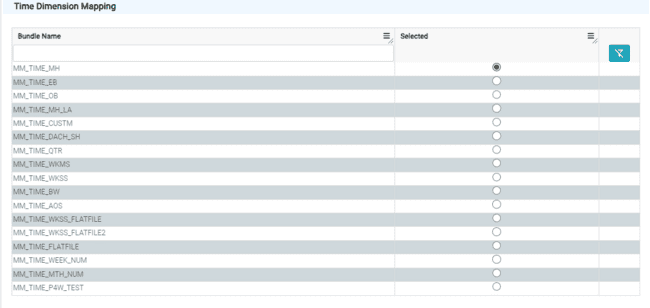
Here the vendor file can be selected for further processing depending of if it’s a three file or single file, further dropdowns get populated.

There are assigned columns that should be mapped mandatory while creating the contract.



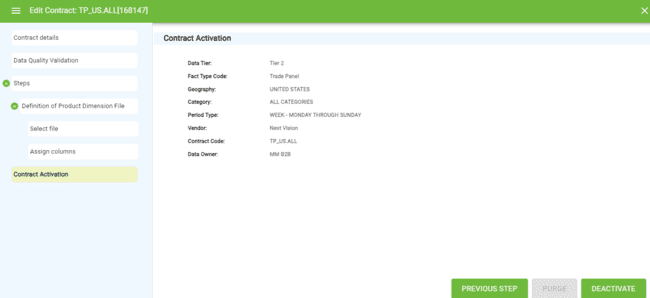
### Time Dimensional Mapping:

Here the bundle value can be selected from the available Bundle names and save it for the contract thats being created.

****

### Contract activation:

Here We can View All the details that are selected earlier for the contract creation, and we can activate/deactivate and purge the contract from this page.



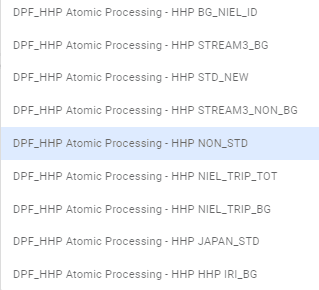
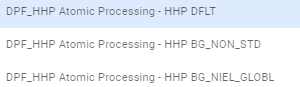
Purge:

When Purge Option is clicked it will Initiate a process in turbine that will remove the records from Fact and Dimension tables from the database against the contract ID.

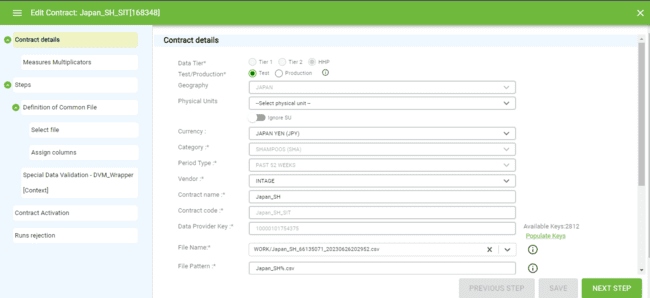
After Purge is completed for the contract, there will be a delete option visible in UI using which the user can delete the Contract data from the Postgres DB.

## HHP Contract Creation:

In HHP Contract Creation there are Twelve Different types of processing flow, The Steps to be followed for all these Graphs are same, the below are the different processing Flow:

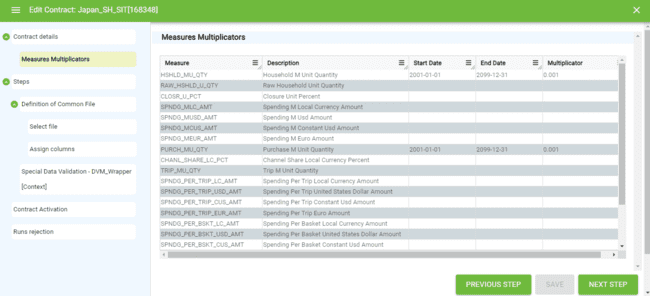
 

**New Contract Page: HHP**

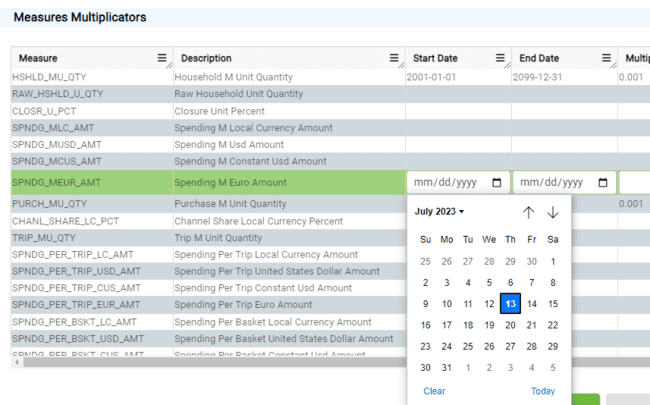


After Filling the Mandatory Fields for a HHP contract and Suitable Processing Flow (Graph), Click on Save button and proceed to next step Which is Measure Multiplicators.

### Measure Multiplicators:

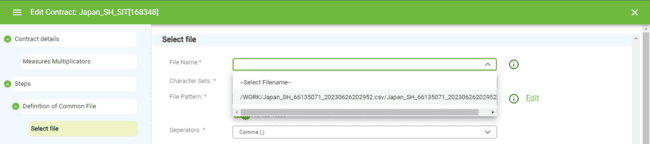


Here There will be Columns necessary for Measures and its description, the Start Date and End date, Multiplicator Value (in Decimals) should be selected from the Dropdown by clicking on the Empty Start Row and End row of the required Measure :

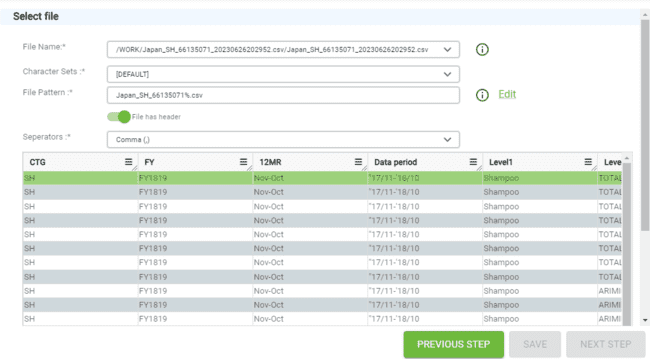


### Select File:

Here the User must Select the Vendor File for the Specific HHP contract which will populate the Data further for Activation of the Contract.



After the required file is selected there will be a preview of the Contents in the selected file and then save, proceed to next Page:

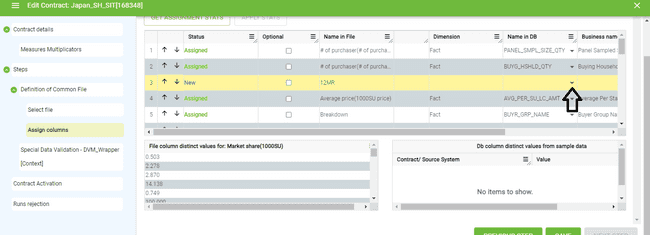


Depending on The delimiter selected the Data’s Will be presented in the UI, and the file pattern should be Proper having a WILDCARD ENTRY (%) so that the selected file can be picked up for the contract.

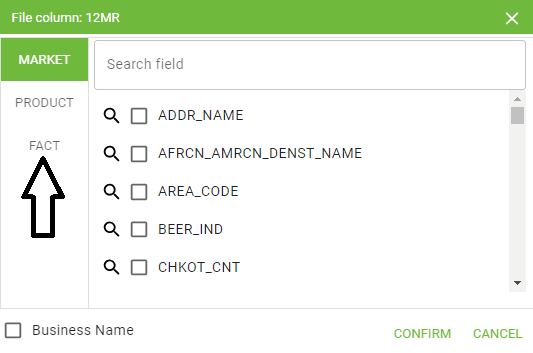
### Assigned Columns (Mappings):

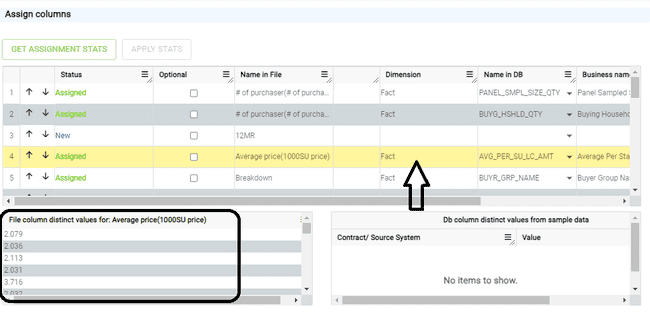
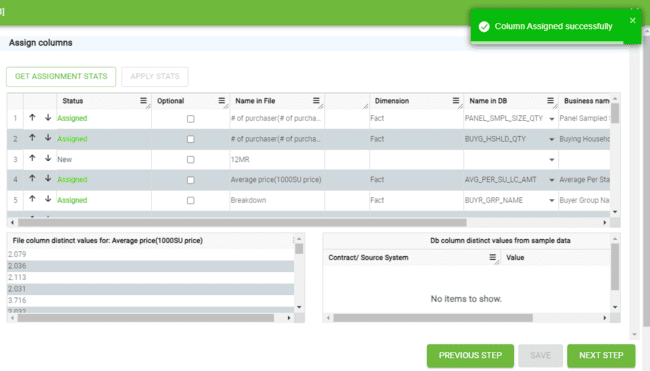
This page contains All the mapping required for the contract that is being created, Name in DB and Dimension should be properly mapped in order to successfully run the Contract.

* To add the required Market, Product and Fact Mappings, click on the Name in DB Arrow mark against a record and a pop-up will appear to select.



* After Clicking on it the Dialog box will look like the below, and required mappings can be selected from that by entering the Business name in the Search Field and click on confirm to apply the mapping:



* To Get a preview of the selected Data clicking on Dimension column, any cell will show a preview below:
* After Assigning the mandatory Mappings click on save and the changes will be saved, clicking on next step will proceed to further Scree

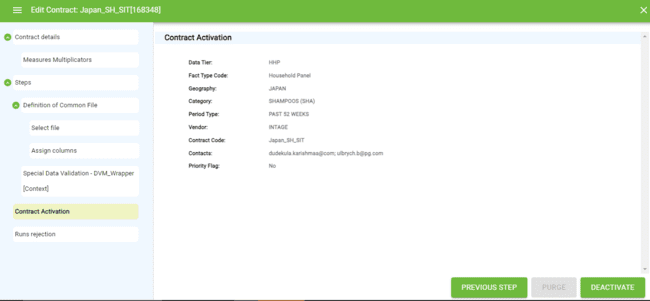
### Special Data Validation:

* Here We will see all the Data validation steps, which will be pre-selected automatically, then save any changes if done and proceed to next step:



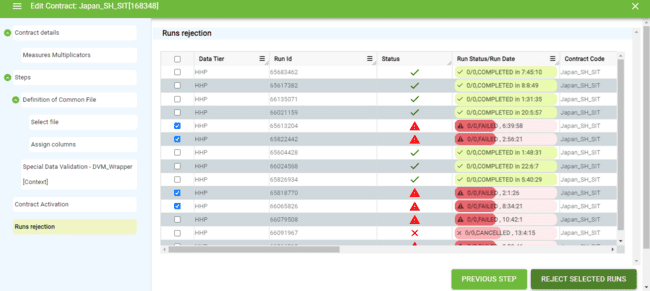
### Contract Activation:

Here We can View All the details that are selected earlier for the contract creation, and we can activate/deactivate, Purge the contract from this page.

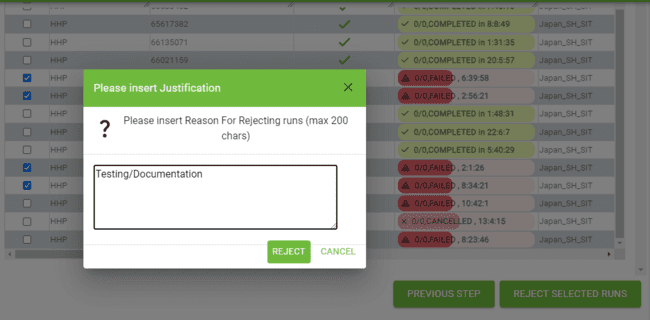


### Runs Rejection:

In this Page we can see All the runs that has been ran for the Specific contract with run details, also the main functionality of this page is it allows the user to “REJECT MULTIPE” runs by selecting the runs that needs to be rejected:



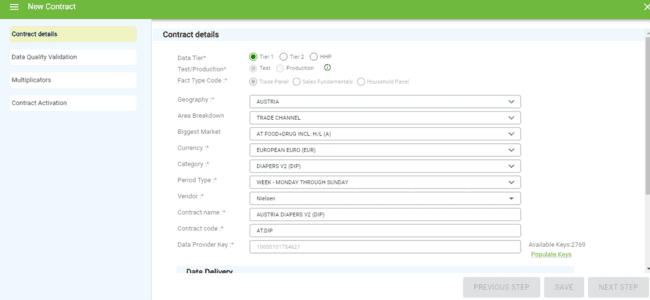
* After the runs are selected, click on the Runs rejected button and there will be a popup with Text Box field asking for the reason to reject the selected runs.
* After clicking on reject, the selected runs will be removed from the UI for the selected contract.

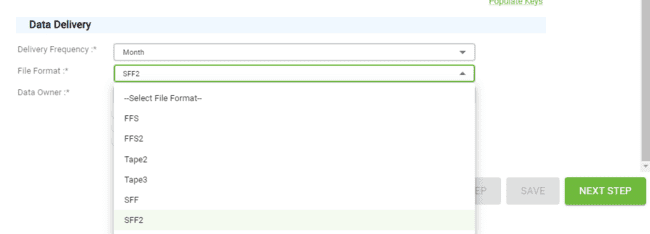


## Tier 1 Contract:

Tier1 means market measurements data that fulfils data standards. Tier1 is applied for EMEA data, For Tier 1 Contract the process flow totally depends only on the FILE FORMAT selected, below are the Six different types of File formats available to select for Tier 1 Contract:

* Tape3
* SFF2
* SFF
* FFS
* FFS2
* Tape 2



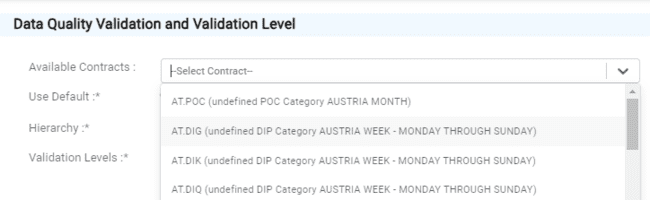
* Below is the File Format Dropdown option for tier 1: 

### Tier-1 Contract Creation:

As Tier1 Does not have any Graph type processing, we have to select the required file format from the dropdown as shown above as per the contract requirement and proceed further steps.

### Data Quality and Validation Level:

This Page contains all details required for the DQ validation for the contract and the Available contracts is the dropdown that shows previously created contracts for the Same selected FILE FORMAT.



* The Use Default button if selected as YES then the page will be only in Read only mode and no edits can be made to the data, but if the option is selected as NO, then All the fields can be edited as per the requirement.
* The Hierarchy Selection Option will show all the Hierarchies that are available for the contract so that as per requirement the level can be selected and proceeded further.
* The Use Default button enables flexibility to the contract creator either to add custom values or keep the default values populated. After entering the required values, click on save and then there will be a Preview of the data below as a tabular format:

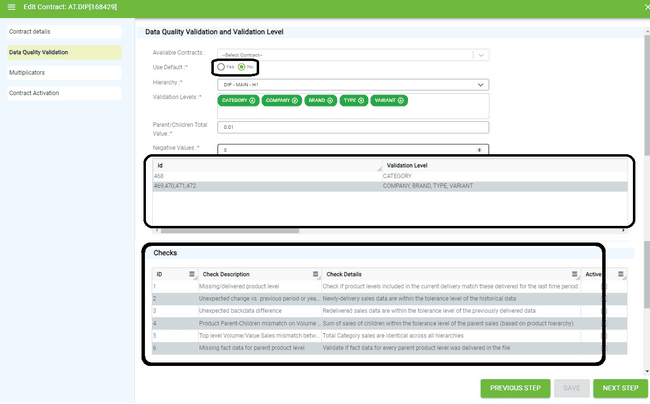
**Hierarchy:**

From the available dropdown values the required one can be selected and saved.

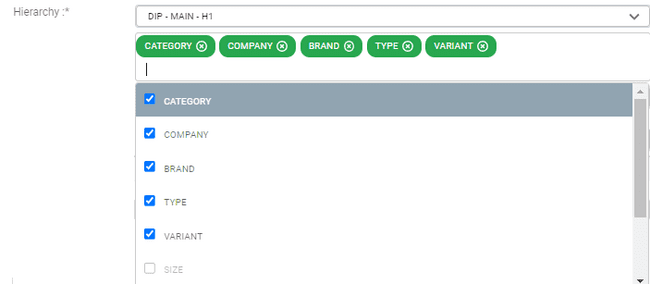


### DQ Validation Page:

After selecting the required values there will be a preview below as follows:



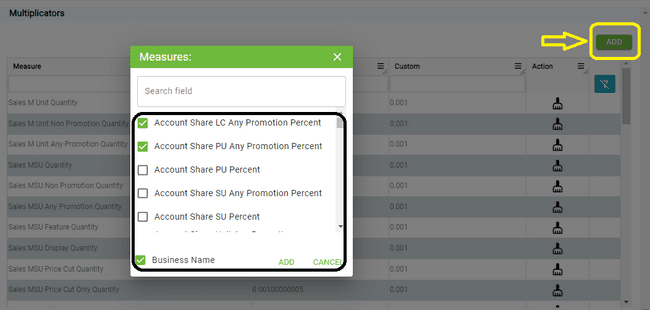
* Validation levels Contain the different levels of business validations that can be selected as per the requirements:



### Multiplicators:

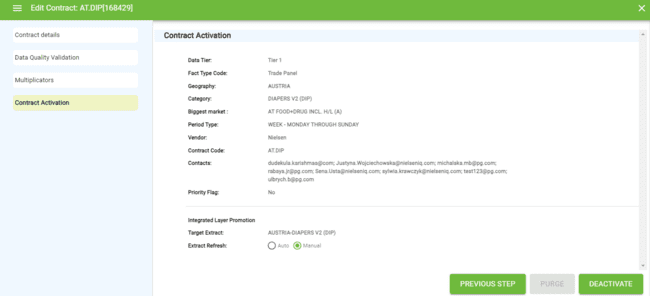
Here we can add all the Measure multiplicators and give the default and custom number for the added rows. After doing this clicking on save and next step will go further.

* After the required Measures are added and the multiplicators are entered, the user will be able to see the selected data on the Multiplicators UI Screen.

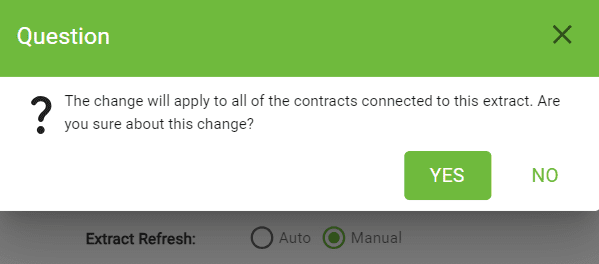


### Contract Activation:

Here We can View All the details of the Tier 1 contract that are selected earlier for the contract creation, and we can activate/deactivate, Purge the contract from this page.



* In this page **Extract Refresh** option can be Set to Auto aand mannual depending on the requirement of the contract, by default it is set as Auto, if its changed to mannual then we get a pop-up asking for a confirmation as shown below :



**API Calls – Contracts**

|  |  |  |
| --- | --- | --- |
| Put/contract/updateHHPMeasureMultiplicator | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/updateHHPMeasureMultiplicator | API calls for updateHHPMeasureMultiplicator of contract data.  (No payload)) |
| Post/contract/updateMeasure | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/ updateMeasure | API calls for updateMeasure of contract data.  (No payload)) |
| Post/contract/updateExtractRefreshType | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/updateExtractRefreshType | API calls for updateExtractRefreshType of contract data.  (No payload)) |
| **Post/** **contract/saveXM** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/saveXML | API calls for saveXM of contract data.  (No payload)) |
| **Post /** **contract/saveValidationChecksStatus** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/saveValidationChecksStatus?contractId=’XYZ’ | API calls for saveValidationChecksStatus of contract data.  (No payload)) |
| **Post /** **contract/saveNewContractDetails** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/saveNewContractDetails | API calls for saveNewContractDetails of contract data.  (No payload)) |
| **Post /** **contract/saveDataQualityValidation** | http://cloudpanel-ui-api.azurewebsites.net/dpf/ contract/saveDataQualityValidation | API calls for saveDataQualityValidation of contract data.  (No payload)) |
| **Post /**  **contract/deactivateContract** | http://cloudpanel-ui-api.azurewebsites.net/dpf/ contract/http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/deactivateContract?contractId=xyz | API calls for deactivateContract of contract data.  (No payload)) |
| **Post /**  **contract/addMeasure** | http://cloudpanel-ui-api.azurewebsites.net/dpf/ contract/addMeasure?contractId= 168558 &measureId=0.01 | API calls for addMeasure of contract data.  (No payload) |
| **Post /**  **contract/activateContract** | http://cloudpanel-ui-api.azurewebsites.net/dpf/ contract//activateContract?contractId= 168558 | API calls for activateContract of contract data.  (No payload) |
| **Post /**  **contract/Export** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/Export?filter=test | API calls for Export of contract data.  (No payload) |
| **Get/ controller/unlockContract** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/unlockContract?contractId=168558 | API calls for unlockContract of contract data.  (No payload) |
| **Get/contract /search** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/search?filter=Test&sort=test&limit=75&page=1 | API calls for search of contract data.  (No payload) |
| **Get/contract /read** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/read?startRow=1&endRow=75 | API calls for read of contract data.  (No payload) |
| **Get/contract/readTresholdsData** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readTresholdsData?contractId=168558 | API calls for readTresholdsData of contract data.  (No payload) |
| **Get/contract/readTargetColumns** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readTargetColumns?contractId=168558&graphStepId=123 | API calls for readTargetColumns of contract data.  (No payload) |
| **Get/contract/readSuggestions** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readSuggestions?contractId=168558&graphStepId=123 | API calls for readSuggestions of contract data.  (No payload) |
| **Get/contract/** **readStepClassTreeNodes** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readStepClassTreeNodes?readOnly=true&isEdit=true&tier=test&factType=test&contractName=Test | API calls for readStepClassTreeNodes of contract data.  (No payload) |
| **Get/contract/** **readMeasures** | http://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readMeasures?contractId=168558 | API calls for readMeasures of contract data.  (No payload) |
| **Get/contract/readHHPMeasuresMultiplicators** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readHHPMeasuresMultiplicators?contractId=168558 | API calls for readHHPMeasuresMultiplicators of contract data.  (No payload) |
| **Get/contract/readFileColumnDistinctValues** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readFileColumnDistinctValues?contractId=168558&graphStepId=123&columnName=test%20 | API calls for readFileColumnDistinctValues of contract data.  (No payload) |
| **Get/contract/** **readDbColumnDistinctValues** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readDbColumnDistinctValues?contractId=168558&tableTypeCode=test%20&columnName=test | API calls for readDbColumnDistinctValues of contract data.  (No payload) |
| **Get/contract/** **readDataProvider** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readDataProvider | API calls for readDataProvider of contract data.  (No payload) |
| **Get/contract/** **readContractOnCntryVndrName** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readContractOnCntryVndrName?countryName=tstcntry&vendorName=tstvndr&tier=tst | API calls for readContractOnCntryVndrName of contract data.  (No payload) |
| **Get/contract/** **readContractDetails** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readContractDetails?contractId=168558 | API calls for readContractDetails of contract data.  (No payload) |
| **Get/contract/** **readAvailableMeasures** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readAvailableMeasures?contractId=168558&measureName=tst%20&measurePhysName=tst | API calls for readAvailableMeasures of contract data.  (No payload) |
| **Get/contract/** **readAssignedColumns** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/readAssignedColumns?contractId=168558&graphStepId=1 | API calls for readAssignedColumns of contract data.  (No payload) |
| **Get/contract/** **purgeContract** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/purgeContract?contractId=168558 | API calls for purgeContract of contract data.  (No payload) |
| **Get/contract/** **populateKeys** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/populateKeys | API calls for populateKeys of contract data.  (No payload) |
| **Get/contract/multiplicator/ search** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/multiplicator/search?filter=tst&sort=ASC&limit=7&page=1 | API calls for multiplicator search of contract data.  (No payload) |
| **Get/contract/** **lockContract** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/lockContract?contractId=168558 | API calls for lockContract of contract data.  (No payload) |
| **Get/contract/** **isContractCodeUnique** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/isContractCodeUnique?contractCode=168558 | API calls for isContractCodeUnique of contract data.  (No payload) |
| **Get/contract/** **getVendorContactList** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/getVendorContactList?contractId=168558 | API calls for getVendorContactList of contract data.  (No payload) |
| **Get/contract/** **getUserLockingContract** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/getUserLockingContract?contractId=168558 | API calls for getUserLockingContract of contract data.  (No payload) |
| **Get/contract/** **getRelatedContracts** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/getRelatedContracts?contractId=168558 | API calls for getRelatedContracts of contract data.  (No payload) |
| **Get/contract/** **getContractCode** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/getContractCode?countryCode=LTI\_TEST&categoryCode=Blades%20 | API calls for getContractCode of contract data.  (No payload) |
| **Get/contract/ deactivateAllOnboardings** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/deactivateAllOnboardings | API calls for deactivateAllOnboardings of contract data.  (No payload) |
| **Get/contract/ checkPattern** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/checkPattern?patternName=tst&contractId=168558 | API calls for checkPattern of contract data.  (No payload) |
| **Get/contract/ canActivateMasterSwitchForTier2** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/canActivateMasterSwitchForTier2?contractId=168558 | API calls for canActivateMasterSwitchForTier2 of contract data.  (No payload) |
| **Delete/contract/** **removeMeasure** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/removeMeasure?multiplierId=1 | API calls for canActivateMasterSwitchForTier2 of contract data.  (No payload) |
| **Delete/contract/** **deleteContract** | https://cloudpanel-ui-api.azurewebsites.net/dpf/contract/deleteContract?contractId=123456 | API calls for deleteContract of contract data.  (No payload) |

# Files

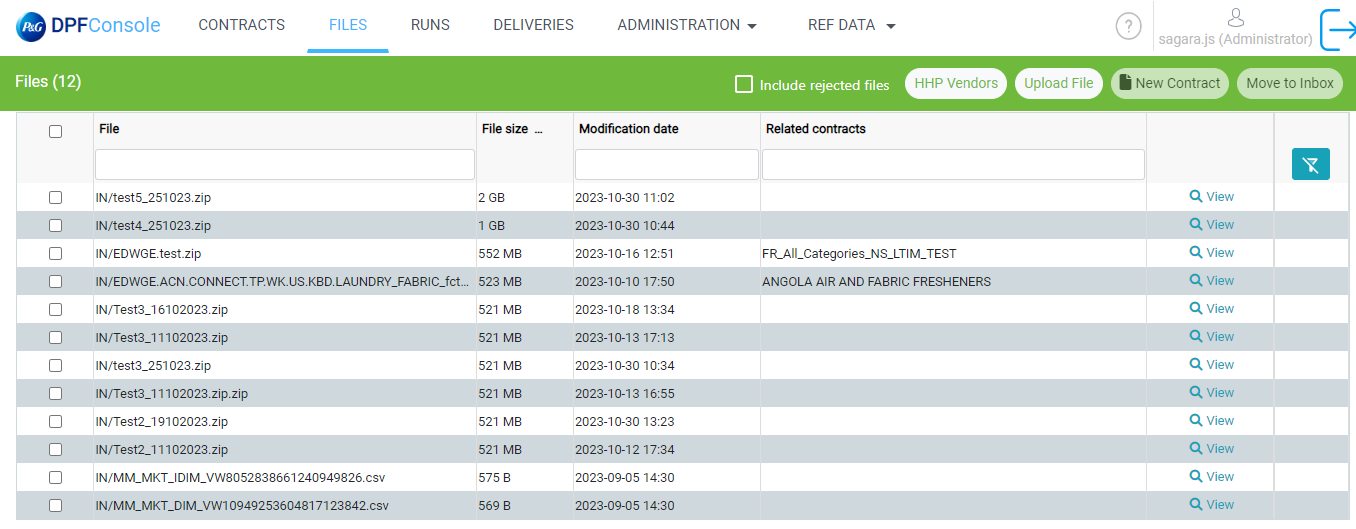
This screen contains following columns:

* Checkbox
* File Size
* Modification Date
* Related Contracts
* Rejected File Indicator
* View

When we click on the files tab initially it will load the files which are coming from IN storage file. In “Azure Blob Storage” we have four different types of storage path IN, REJECT, WORK and ARCH. Files screen only utilizing IN (Inbox folder) and REJECT (All rejected files will here) folder. Initially Files loads the Inbox (IN) folder files.

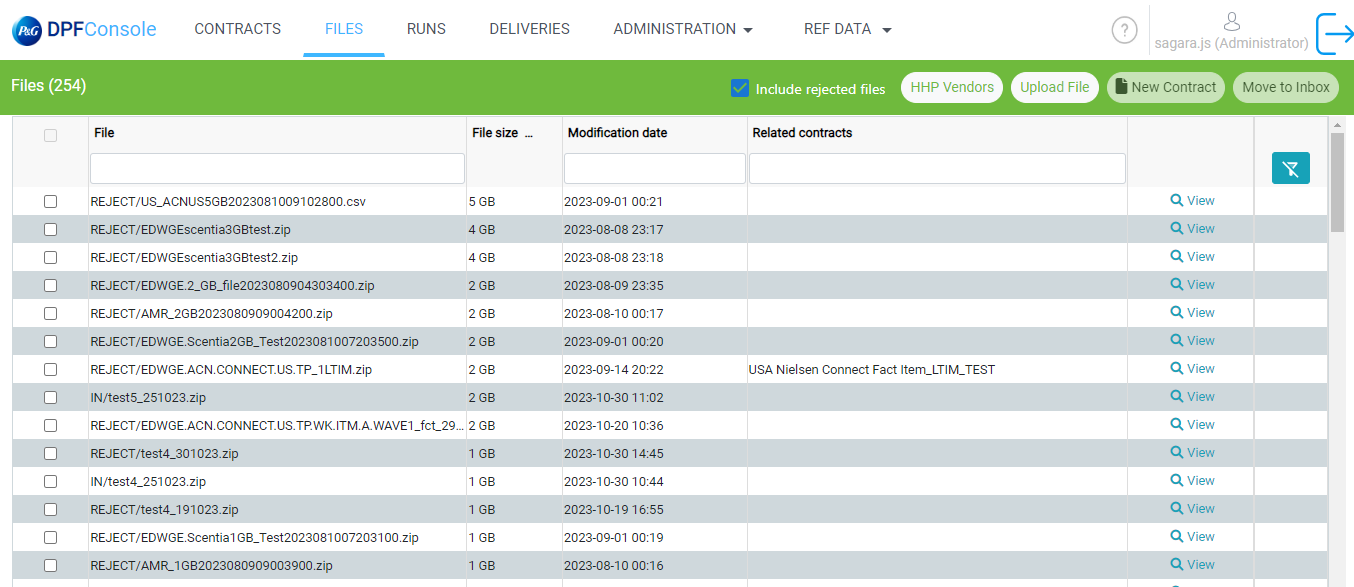
IN related files are the files which are yet to be processed.

If there are any contract assigned with some file pattern and that pattern matches with the any of these files, then from the IN-folder turbine will start processing and it will remove that file from IN and file will moved to WORK folder with file name appended with Processing ID and timestamp. After successful processing the file will get moved to ARCH folder.



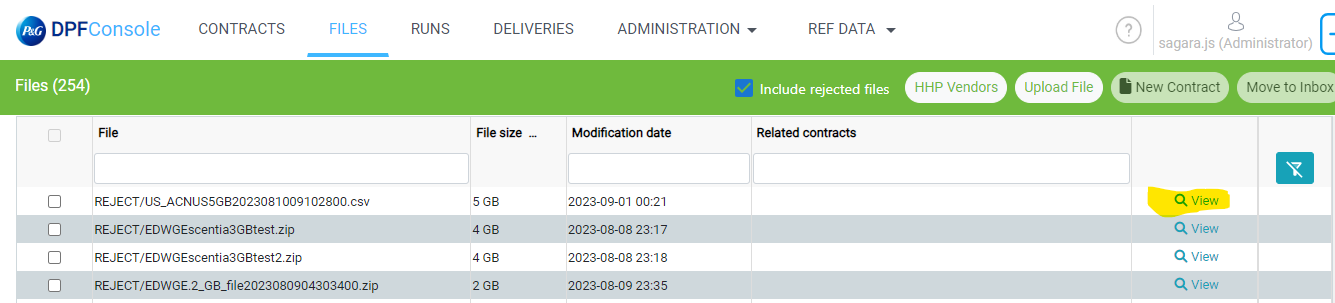
## Include rejected files:

Files which are available in REJECT folder will be shown as checked in Reject file indicator column, you can select that particular file and that enables MOVE TO INBOX button and once you click on that button it moves the file to IN folder, if the matching pattern found for that file then TURBINE will take up that file for the Processing, or else immediately file will be moved back to REJECT folder as per TURBINE architecture



## View

When we click on view, we can see the content of that file.



Once we click on view one page will open in that page all the content of file will display

Please check below screenshot.

A screenshot of a computer

Description automatically generated with medium confidence

In this page we can see File Name, File Has Header checkbox and Separators.

If we uncheck that File has header checkbox column name will replace by column 0, column 1. Please check below screenshot.

A screenshot of a computer

Description automatically generated

And there is Separators dropdown also in this dropdown there are four options Comma, Semicolon, Pipe, Tabulation. When we select any of these options API call will trigger, and we will get in that format.

A screenshot of a computer

Description automatically generated with medium confidence

Note:- This View option will not show the data for the Master file means where it will have 3 files inside of it.

## HHP Vendor Button

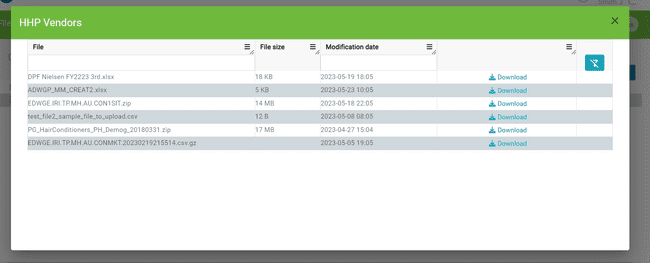
A screenshot of a computer

Description automatically generated

When we click on HHP Vendor button it will open one popup in that Whatever the files which contract has been assigned with HHP will display. In Contract we have three Data Tiers Tier1, Tier2 and HHP. In main screen we are loading all Data Tiers files.

In that table you can see File, File Size, Modification Date and Download Columns.

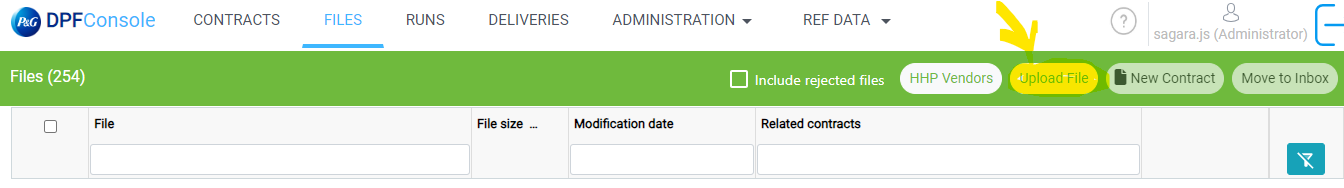
This Pop Up will open once we click on the HHP Vendor button.



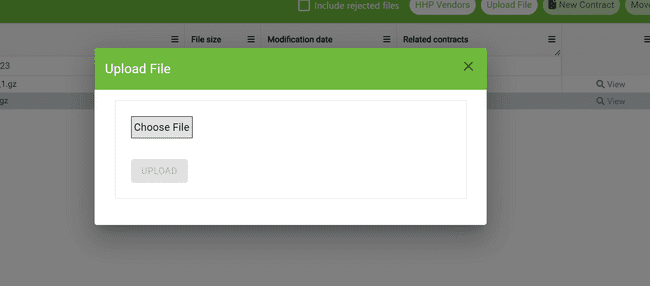
In Above screenshot there is download on click of that file will download.

## Upload File Button

This button is there in Header we can see in below screenshot.



On click of this button one popup will come using this we can upload file. We can upload single as well as multiple files. Using this choose file button we can select files from the local which we have to upload.



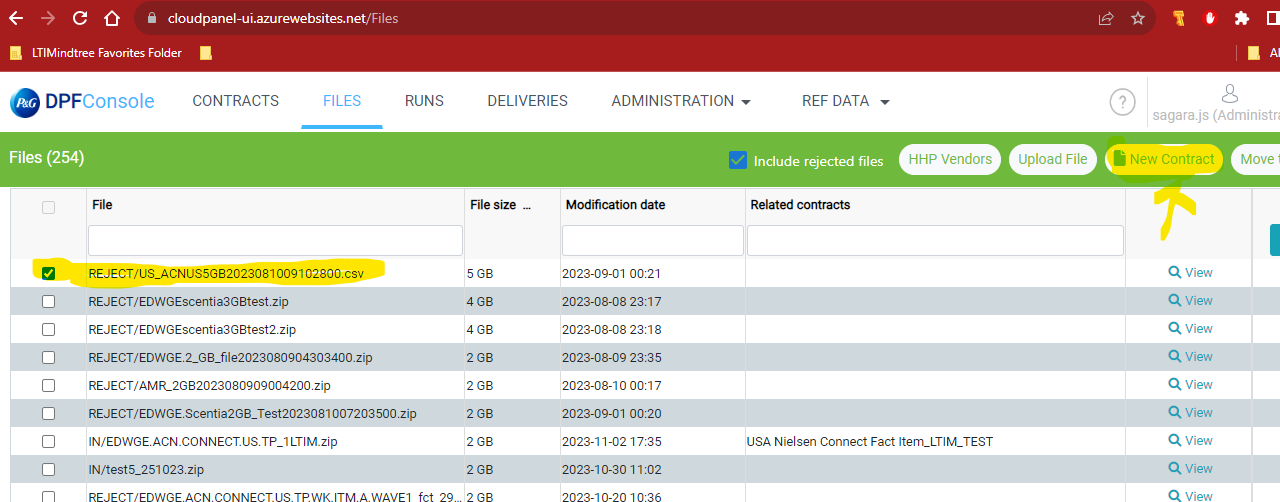


Once the file was uploaded it will move to IN folder in “Azure Blob Storage” and we have sync job once we upload the file it will sync to our Postgres DB in every ten seconds.

And there is turbine strategy if uploaded file doesn’t match with any of the contract if it is not in standard format, it will immediately move that file in REJECT folder. (With some delay it should display in IN or REJECT folder).

## New Contract

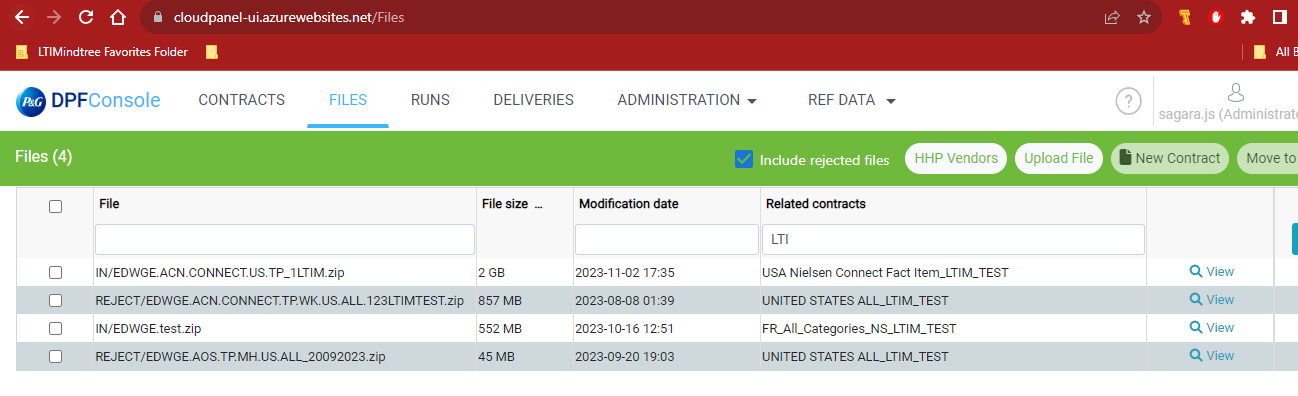
This Button will only be enabled when there is no related contract for that file. Please check below screenshot.by Default, this button will be disabled.



When we select a file which doesn’t have a related contract. New Contract option will be enabled, please check below screenshot. This will enable for that file which have empty related contract.

In the Below screenshot file is selected “ REJECT/EDWGE.ACN.CONNECT.TP.WK.US.ALL.123LTIMTEST.zip” .

Which have empty “Related Contracts”.



After selection of checkbox(file) which don’t have related contract “New Contract” will be activated.

Then after clicking on “New Contract” button it will open one popup window that is new Contract creation page. We must fill in all the required fields for creating contract and we won’t have option to select file name here, automatically file name will be generated based on the files selected in files screen.

A screenshot of a computer

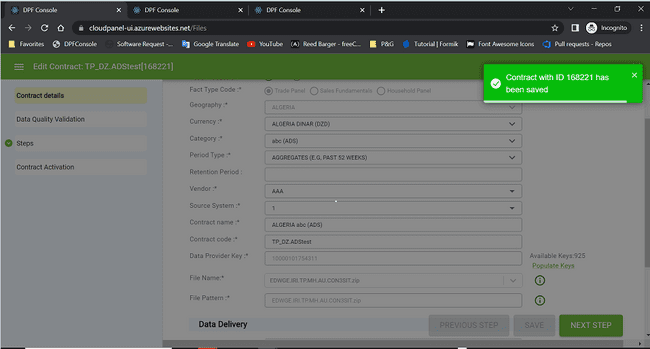
Description automatically generated

After filling in the form “Save” button will be activated.

A screenshot of a computer

Description automatically generated

After clicking on “SAVE” button the new contract will be created. This popup below will come once we click on “SAVE” button.



## Move To Inbox:

To move files from REJECT folder to IN folder for processing.



By default, Move to Inbox option is disabled. When the reject file is selected then that option will be enabled.

Please check below screenshot it is enabled on selection of rejected checkbox file.



This file moved to IN folder then turbine will do all validations check and if that file does not match any of the file pattern, then it will be moved back to REJECT folder.

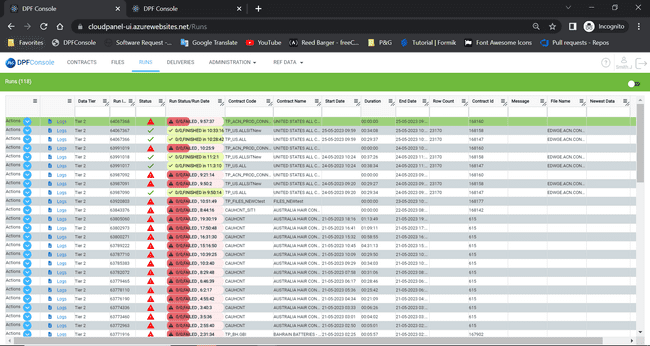
**API calls Files:**

|  |  |  |
| --- | --- | --- |
| **/fileStepClass/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/fileStepClass/search?page=1&limit=75&filter=(rejectedFileId~eq%3A0)** | **API call for search files data(No PayLoad)** |
| **/fileStepClass/search (HHP vendor)** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/fileStepClass/search?page=1&limit=75&filter=(datatier~like%3Ahhp)** | **API call for search files (HHP Vendor)data(No PayLoad)** |
| **/fileStepClass/upload** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/fileStepClass/upload** | **API call for Upload files**  **Payload require.** |
| **/dictionary/readFactTypeCodes** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/dictionary/readFactTypeCodes** | **API call for New Contract**  **(No Payload)** |
| **/fileStepClass/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/fileStepClass/search?page=1&limit=75** | **API call for include rejected files data(No PayLoad)** |
| **/fileStepClass/moveToInbox** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/fileStepClass/moveToInbox?fileName=”filename”** | **API call for include rejected files data:**  **PayLoad: fileName:”filename”** |
| **/fileStepClass/readFileData** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/fileStepClass/readFileData?fileName=DPFNielsenFY2223\_3rd.xlsx&separator=;&fileHasHeader=true&fullFilePath=turbinev1/IN/DPFNielsenFY2223\_3rd.xlsx** | **API call for include rejected files data(No PayLoad)**  **Query String Parameter:**  **fileName: “filename”**  **saparator: ;**  **fileHasHeader: true**  **fullFilePath: “turbine1/IN/filename”** |

# Runs

This screen contains following columns:

* Actions
* Logs
* Data Tier
* Run Status/Run Date
* Status
* Run ID
* Contract ID
* Contract Name
* Contract Code
* Message
* File Name
* Row Count
* Newest Data
* Start Date
* End Date
* Duration



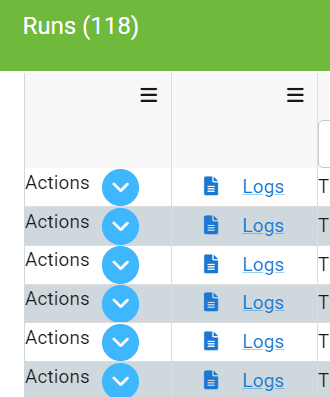
## Auto Refresh:

Auto refresh toggle button helps the page to fetch all the latest records and it is present on top right of the page, when this toggle button is clicked/enabled the page will refresh automatically at a 10 second interval.



## Logs:

It will redirect to turbine for that run id where we clicked.



Here let’s click on logs for run id 64286885:

A screenshot of a computer

Description automatically generated

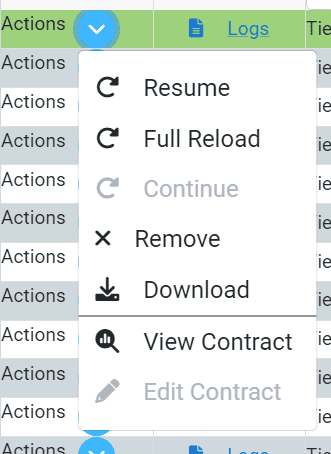
It redirected to turbine page for run id 64286885:

A screenshot of a computer

Description automatically generated

## Action Dropdown:

Every row/record will have actions button as its first column, it helps to perform various actions on selected record.



The following Options can be seen and performed under Actions dropdown:

* Resume
* Full Reload
* Continue
* Remove
* Download
* View Contract
* Edit Contract

### Resume:

It will start processing from the last milestone. Only failed status runs will be enabled for all data tiers and DQ issue failed status for HHP tier. It will start processing for the same run id and change the state to initialized or waiting state.

A screenshot of a computer

Description automatically generated with medium confidence

Before Clicking Resume Action state of that Run Id status is Failed

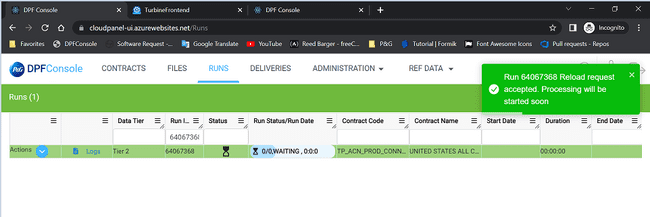


In Turbine also it is in a failed state.

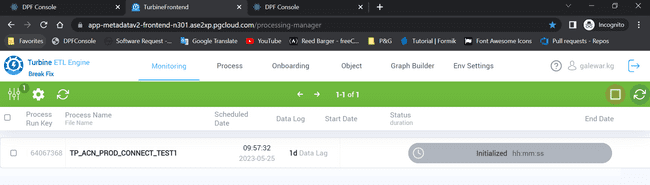
A screenshot of a computer

Description automatically generated with medium confidence

After Clicking on Resume Action state is changed in DPF console from Failed to Waiting.



In Turbine also state was changed from failed to initialize.



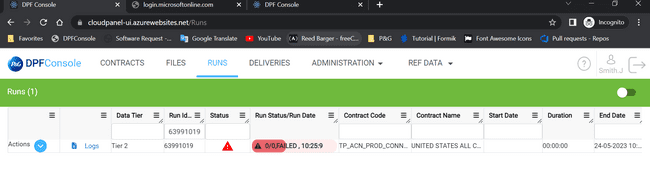
### Full Reload:

It will restart the process from the beginning and this option will be enabled for Failed and Finished status for all data tiers and DQ issue failed status for HHP tier. (It will create a new run id in turbine).

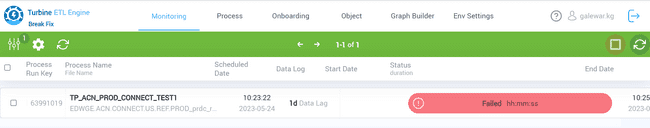
A screenshot of a computer

Description automatically generated with medium confidence

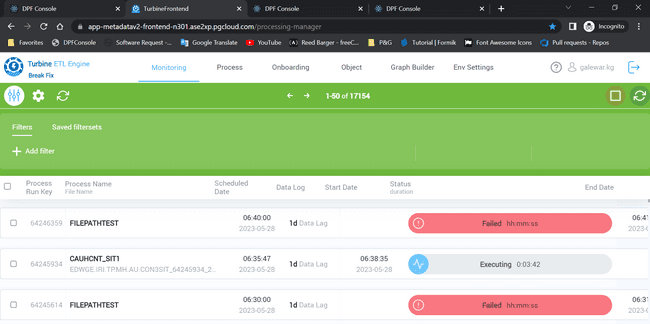
Before Clicking Full Reload Action state of that Run Id is Failed



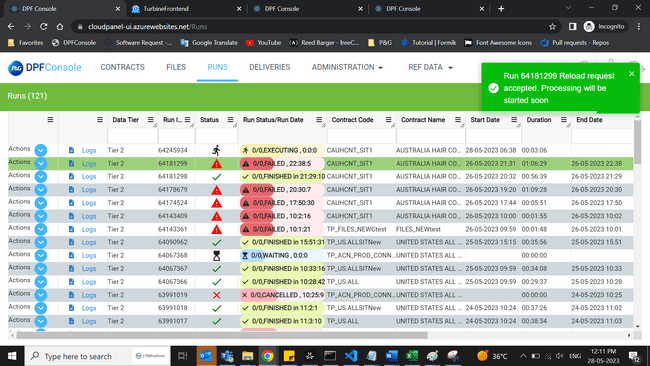
In Turbine also it is in a failed state.



Before Clicking on Full reload in Turbine:



Before clicking on Full reload in DPF console:



After clicking on the full reload action on top new run id (64246366) is created. Please check below screenshot.

A screenshot of a computer

Description automatically generated

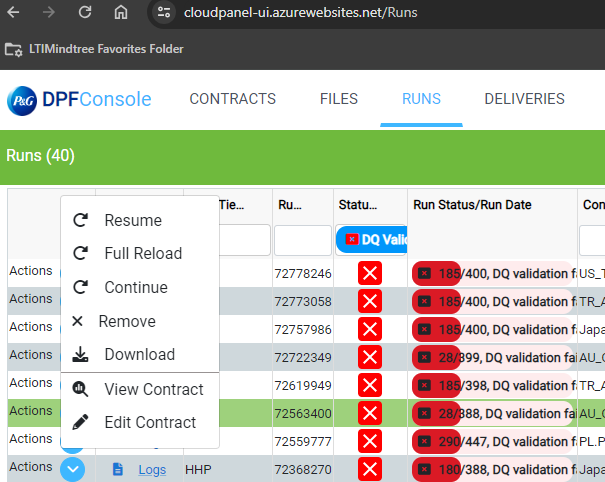
After clicking on full reload in turbine new run id (64246366) is created. Please check below screenshot

A screenshot of a computer

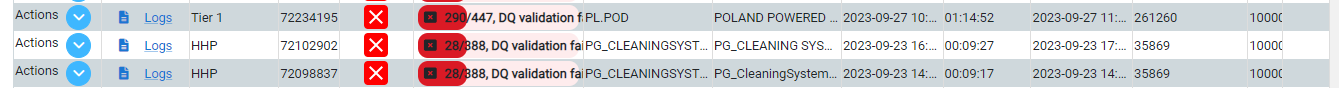
Description automatically generated

### Continue:

Processing will be continued starting from next step and it will Ignore DQ Issue via pass IGNORE\_DQ\_ISSUE param in payload to run-parameter. This option will be enable only in DQ\_Issue status for all data tiers. It will continue process on same run id then status will be changed to initialized or waiting state.



Before Clicking Continue Action state of that Run Id status is DQ validation failed



In Turbine also it is in same DQ\_Issue state.

A screenshot of a computer

Description automatically generated

After Clicking on Continue Action state is changed in DPF console from DQ validaton failed to Waiting.

A screenshot of a computer

Description automatically generated

In Turbine also state was changed from DQ\_Issue to waiting/initialize.

A screenshot of a computer

Description automatically generated

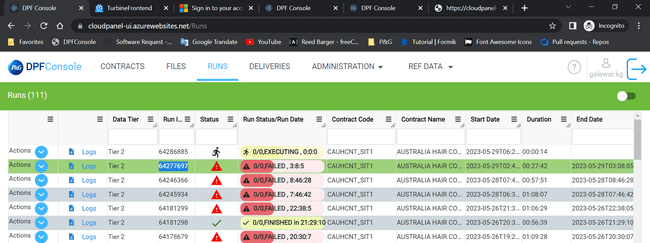
### Remove:

After clicking on the remove button. The selected Process will be canceled, and its Run will be removed. it will not be visible in DPF console. Remove button will be enabled/clickable only in Failed, Preparing, DQ\_ISSUE state of the run.

A screenshot of a computer

Description automatically generated with medium confidence

Before clicking on remove action in DPF console we can see <runid> in screen and status also failed:



Before clicking on remove action in Turbine we can see run id 64277697 in screen and status also failed:

A screenshot of a computer

Description automatically generated

Once we click on remove action item, we will get one notification “Runs Successfully Removed”:

A screenshot of a computer

Description automatically generated

After clicking on remove action in DPF console we can see run id 64277697 is not in screen entire row was deleted:

A screenshot of a computer

Description automatically generated

And in turbine status was updated from failed to be cancelled:

A screenshot of a computer

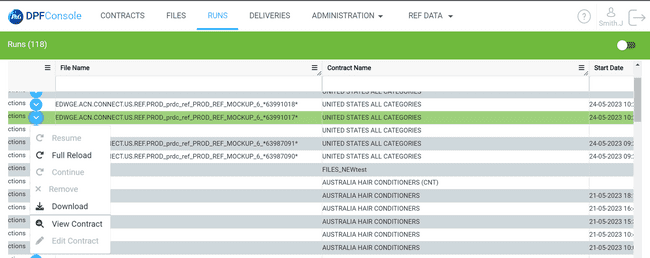
Description automatically generated

### Download:

Vendor File against which run was executed will be downloaded.

A screenshot of a computer

Description automatically generated with medium confidence



A screenshot of a computer

Description automatically generated

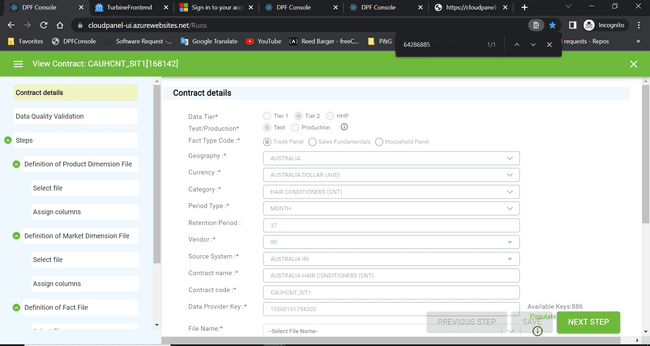
### View Contract:

With the help of this action, we can view that particular contract depends on contract id what you selected.

A screenshot of a computer

Description automatically generated with medium confidence

We can be able to view contract on clicking on view contract action (same as contract page):



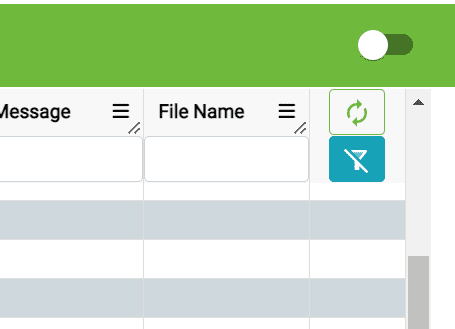
### Edit Contract:

With the help of this action, we can be able to Edit that particular contract depends on contract id what you selected.



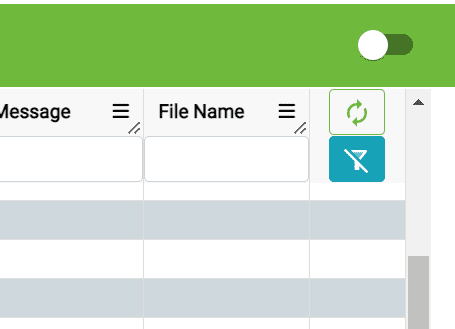
## Refresh Button

It will refresh the page on each click. This button is just below the auto refresh toggle button.



## Clear Filter Button:

This button will clear all filters whatever we applied.



**API calls-Runs:**

|  |  |  |
| --- | --- | --- |
| **/run/resume** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/run/resume** | **API call for Resume action.**  **(Require runId as payload)** |
| **/run/fullReload** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/run/fullReload** | **API call for fullreload action.**  **(Require runId as payload)** |
| **/run/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/run/search?page=1&limit=75&sort=-runId** | **API call for search runs data(No PayLoad)** |
| **/run/remove** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/run/remove** | **API call for Remove action.**  **(Require runId as payload)** |
| **/run/continue** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/run/continue?runId=63688344** | **API call for Continue action.**  **(Require runId as payload)** |

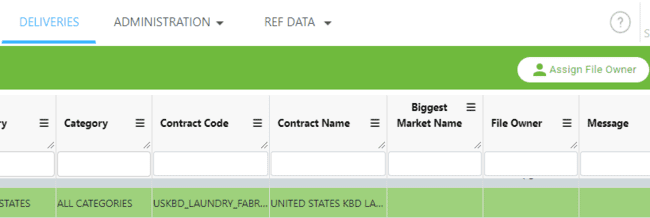
# Deliveries:

This screen contains all the delivery related functionalities and options for all the contracts that are created in the Cloud panel, this screen contains following columns:

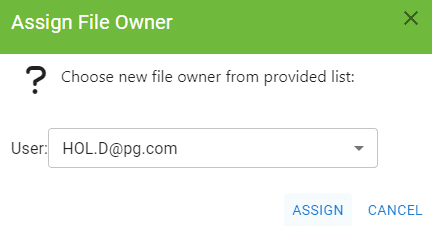
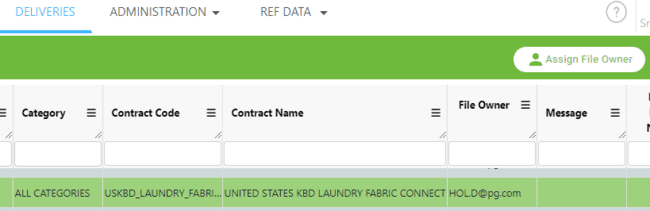
* Actions
* Logs
* Process
* Run Status/Run Date
* Status
* Run ID
* Contract ID
* DQ validation
* Business Validation Report
* Delivery Date
* SLA Date
* Scheduled Date
* Priority
* Contract Status
* ID
* Region Hub
* Country
* GBU/Sub-GBU
* Category
* Contract Code
* Contract Name
* Biggest Market name
* File Owner
* Message
* File Name
* Row Count
* Newest Period
* Vendor
* Fact Type
* Last Processed
* Data Tier
* Last Activity
* Master Switch
* Auto Validation

## Assign File Owner:

To assign a file owner for any specific record/delivery, this option will be helpful as this will show all the file owners present as a dropdown for selection and then confirm, save it.

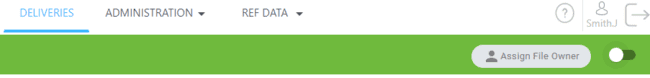


After selecting a record and clicking on assign file owner the Pop-up will appear as shown below and after assigning the owner (Here: [HOL.D@pg.com](mailto:HOL.D@pg.com)) the specific owner will be assigned to that delivery record successfully.

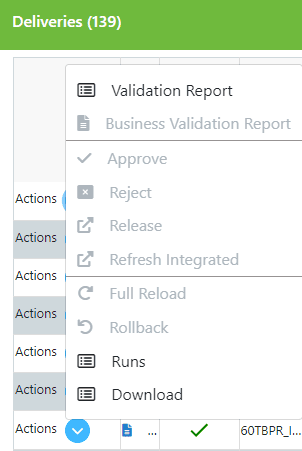
## Auto Refresh:

Auto refresh toggle button helps the page to fetch all the latest records and it is present on top right of the page near Assign file owner option, when this toggle button is clicked/enabled the page will refresh automatically at a 5 second interval.



## Actions dropdown button:

Every row/record will have an actions button as its first column, it helps to perform various actions on selected delivery record.



The following Options can be seen and performed under Actions dropdown:

* Validation Report
* Business Validation
* Approve
* Reject
* Release
* Refresh Integrated
* Full Reload
* Rollback
* Runs
* Download

## Validation Report:

This Functionality helps the user to download a detailed validation report in excel format of the selected delivery run, validation report generated for selected vendor will contain all details.

A screenshot of a computer

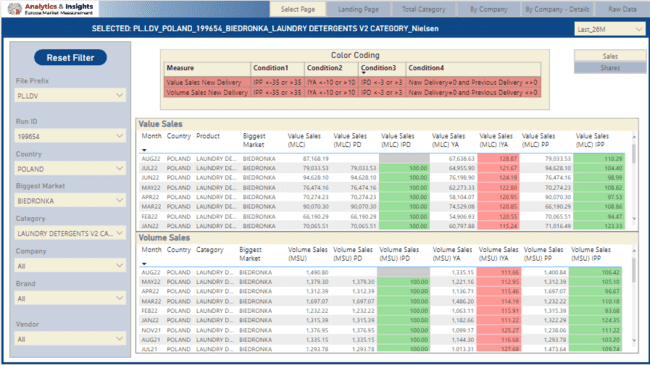
Description automatically generated

A screenshot of a computer

Description automatically generated

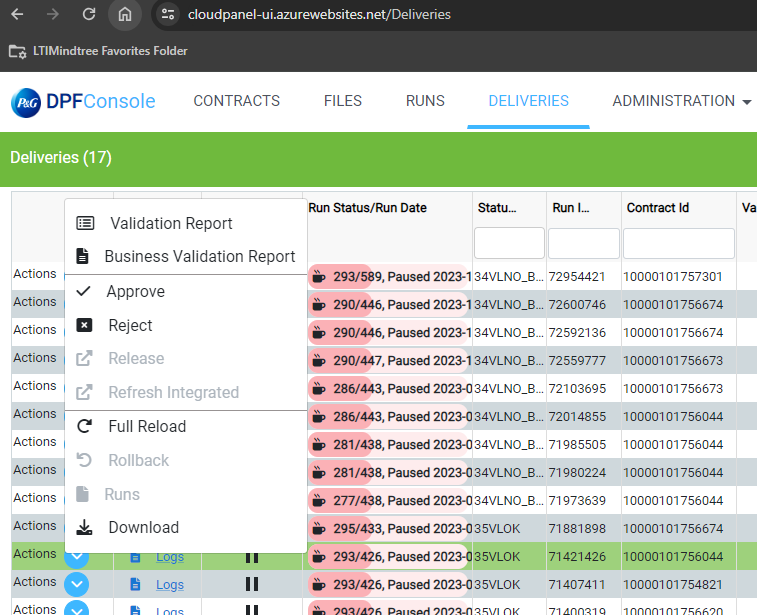
## Business Validation:

This Functionality helps the user to view/generate a detailed Business validation report of the selected delivery, business report generated for the selected delivery will open a POWER-BI report page.



## Approve:

This Functionality helps the user to Approve the selected delivery, after clicking on Approve, Process will be resume with confirmation message that” Delivery {processRunKey} has been approved. Processing will be started soon”.



Before clicking on remove action in cloud console we can see run id 72954421 in screen and status also Paused

A screenshot of a computer

Description automatically generated

Before clicking on remove action in turbine we can see run id 72954421 in screen and status also DQ\_Issue

A screenshot of a computer

Description automatically generated

After Clicking on Approve Action status is changed in DPF console from Paused to Waiting.

A screenshot of a computer

Description automatically generated

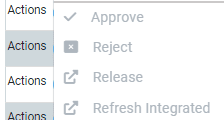
After Clicking on Approve Action status is changed in Turbine from Paused to Waiting/Initialized.

A screenshot of a computer

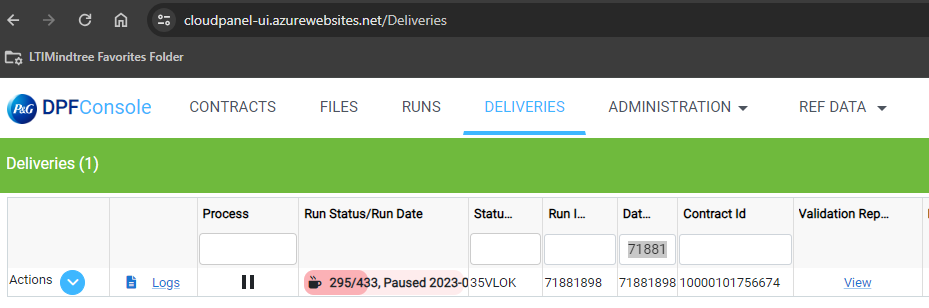
Description automatically generated

## Reject:

This Functionality helps the user reject the selected delivery, after reject option is clicked, the following message is displayed “Delivery {ProcessRunKey} has been rejected”.



Before clicking on Reject action in cloud panel UI console we can see run id 71881898 in screen with Paused status



Before clicking on reject action in turbine we can see run id 71881898 in screen with DQ\_Issue status

A screenshot of a computer

Description automatically generated

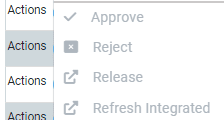
After Clicking on reject Action status is changed in cloudpanel ui console from Paused to Finished with 70REJD delivery phase status.

A screenshot of a computer

Description automatically generated

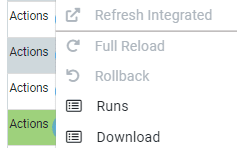
## Release:

This Functionality helps to trigger the integrated contract and release the selected delivery for a tier 1 contract and process status should be paused, the following message is displayed after this option is clicked “Release request accepted. Processing will be start soon”.



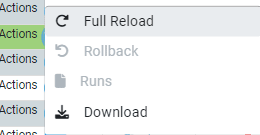
## Refresh Integrated:

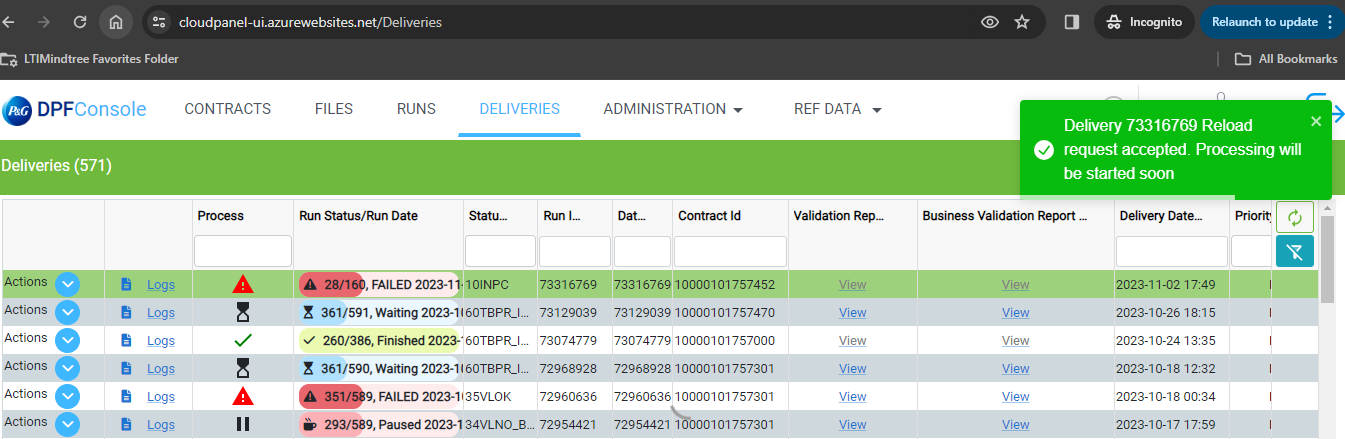
This Functionality helps to trigger the integrated contract for a tier 1 contract and the following message is displayed after this option is clicked “refresh Integrated request accepted. Processing will be start soon”.

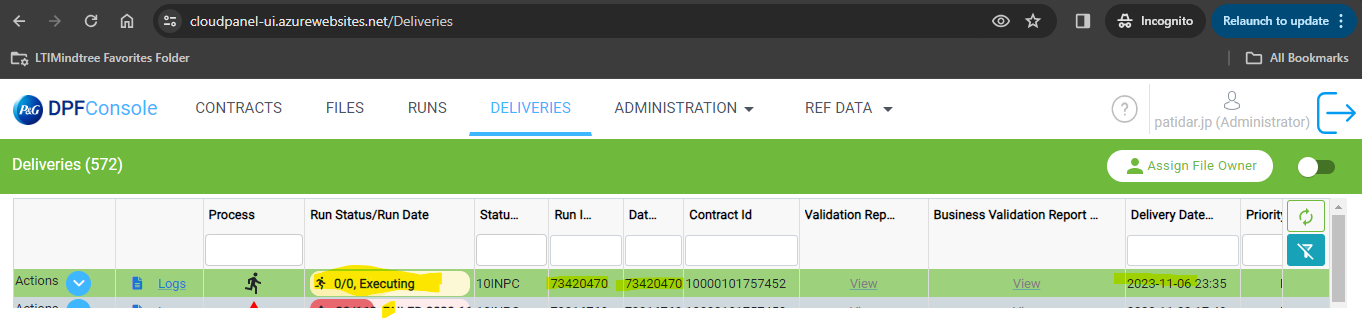


## Full Reload:

This Functionality restarts processing from scratch with file from given run. It means that if user wants to reload contract with previous month data – it is possible. There is need to select row with valid information and within actions button select “Full Reload”.

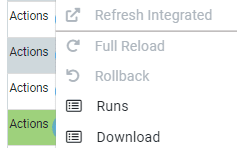






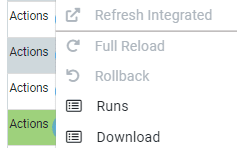
## Rollback:

This Functionality rejects the current run and whatever the data is pre-published will be reverted back and this option only applies to tier1 contract, and it must be clicked before clicking the refresh integrated option.



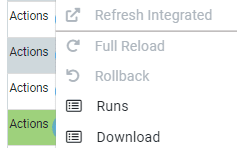
## Runs:

This Functionality redirects to RUNS page and shows all the runs related to the specific contract.



## Download:

This Functionality helps the user to download Vendor File against which run was executed.



**API Calls – Deliveries**

|  |  |  |
| --- | --- | --- |
| Post /delivery/reject | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/reject | API calls for reject of Deliveries data.  (No payload)) |
| Post /delivery/processRollBack | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/processRollBack | API calls for processRollBack of delivery data.  (No payload)) |
| Post /delivery/processRefreshIntegrated | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/processRefreshIntegrated | API calls for processRefreshIntegrated of delivery data.  (No payload)) |
| **Post /delivery/ processMannualRelease** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/processManualRelease | API calls for processMannualRelease of delivery data.  (No payload)) |
| **Post /delivery/fullReload** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/fullReload | API calls for fullReload of delivery data.  (No payload)) |
| **Post /delivery/approve** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/approve | API calls for approve of delivery data.  (No payload)) |
| **Post /delivery/addfileOwners** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/addfileOwners | API calls for addfileOwners of delivery data.  (No payload)) |
| **Get/ delivery/search** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/search?limit=75&page=1 | API calls for search of delivery data.  (No payload)) |
| **Get/ delivery/processDetails** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/processDetails | API calls for processDetails of delivery data.  (No payload)) |
| **Get/ delivery/deliveryPhases** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/deliveryPhases | API calls for deliveryPhases of delivery data.  (No payload)) |
| **Get/ delivery/deliveryList** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/deliveryList?startRow=0&endRow=75 | API calls for deliveryList of delivery data.  (No payload)) |
| **Get/ delivery/businessValidationReport** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/  businessValidationReport?dlvryId=xxx&contractId=xxx | API calls for businessValidationRepor of delivery data.  (No payload)) |
| **Get/ delivery/assignfileOwners** | http://cloudpanel-ui-api.azurewebsites.net/dpf/delivery/assignfileOwners | API calls for assignfileOwners of delivery data.  (No payload)) |

# Administration.

In administration we have six sub modules.

Administration -> Chains

Administration -> Application Logs

Administration -> Security Scopes

Administration -> Source System

Administration -> Vendors

Administration -> Integrated Contract

A screenshot of a computer

Description automatically generated with medium confidence

## Chains.

When we click on the chains tab it will redirect to turbine graph screen.

## Application Logs.

In application logs we have six columns

* Contract
* Contract Name
* User
* Action
* Date
* Message

In other pages (contract, , files). When we are creating contract, application log gets created and those are stored in a database table. In “users” column we can see which user created particular contract and in “message” column we will see massages related to that contract whether it is deleted or modified.

A screenshot of a computer

Description automatically generated with medium confidence

**API calls Application logs**

|  |  |  |
| --- | --- | --- |
| **GET**  **/** **applicationLog/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/applicationLog/search?page=1&limit=75** | **API for search-(payload not required)** |

## Security Scopes

In security scopes we have Group and Filters.

A screenshot of a computer

Description automatically generated

**New Scope**

We can see this button in the header.

A picture containing text, font, line, screenshot

Description automatically generated

once we click on the “New Scope” button it will opens one popup. You can see below screenshot.

A screenshot of a computer

Description automatically generated with medium confidence

In this popup we have “Group Name” input box it is active directory group we can enter here group name. And then Data Tier, Geography, Category, Data Owner, Additional Contracts these all fields are in contract.

When we select “Data Tier”. On the right we can see “Contract Matching” section where all-matching contracts will be displayed. Refer below screenshot where list of contracts are fetched for the selected Date Tier : “Tier 1”.

A screenshot of a computer

Description automatically generated with medium confidence

Once we click on “save” button default value for “Geography”,” Category”,” Data Owner” will be saved. And whatever we saved, it will be listed in our main screen Please check below screenshot.

A screenshot of a computer

Description automatically generated

Once we click on the row in right side, then on left side of screen under “Contract matching” section all matching contracts will load.

The Edit button in contract is not enabled for everyone as authorization is implemented. Suppose you have View access, then as a viewer you can’t be able to edit that contract. Based on security scope you can be able to view or edit the contract.

A screenshot of a computer

Description automatically generated with medium confidence

**Edit**

You can edit security scope by this button.

A screenshot of a computer

Description automatically generated with medium confidence

Once we click on the edit button one popup will come here, we can edit existing details.

A screenshot of a computer

Description automatically generated with medium confidence

**Delete**

A screenshot of a computer

Description automatically generated with medium confidence

Once we click on the delete button it will delete that particular row. This entry will be deleted from database.

**API calls for Security Scope**

|  |  |  |
| --- | --- | --- |
| **GET**  **/administrations/scope/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/scope/search?page=1&limit=75** | **API for search-(payload not required)** |
| **GET**  **/administrations/scope/readContractsForScope** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/scope/readContractsForScope?groupName=EDW-KKHS** | **API for readContractsForScope (payload required in queryParameter groupName=<groupName>)** |
| **GET**  **/dpf/contacts/readRegions** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/contacts/readRegions** | **API for Read Regions-(payload not required)** |
| **GET**  **/dpf/dictionary/readCategoriesWithId** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/dictionary/readCategoriesWithId** | **API for Read readCategoriesWithId -(payload not required)** |
| **GET**  **/dpf/dictionary/readDataOwners** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/dictionary/readDataOwners** | **API for Read readDataOwners (payload not required)** |
| **GET**  **/dpf/dictionary/readContracts** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/dictionary/readContracts** | **API for Read readContracts -(payload not required)** |
| **GET**  **/administrations/scope/readScopePossibleContracts** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/scope/readScopePossibleContracts?categoryName=&countryName=&owner=&dataTier=Tier%201** | **API for Read readScopePossibleContracts -(payload in queryParameter :dataTier, categoryName, countryName, owner** |
| **POST**  **/administrations/scope/saveScope** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/scope/saveScope** | **API for saveScope**  **(payload required in request body :**  **{"grpName": "","cntryList": [],"categList": [],"cntrtList": [],"dataOwnerList":[]"dataTier": ["Tier 1"]})** |
| **DELETE**  **/administrations/scope/deleteScope** | [**https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/scope/deleteScope?groupName=EDW-KKHS**](https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/scope/deleteScope?groupName=EDW-KKHS) | **API for deleteScope -(payload required in queryParamter : groupname = <groupname>** |

## Source System

In Source System screen we have two columns Name and Description.

A screenshot of a computer

Description automatically generated with medium confidence

**New Source System**

This button will be present in the Header on the top right of the UI.

A green rectangle on a white background

Description automatically generated with low confidence

On clicking of this button one popup will come with heading of “New source system”.

A screenshot of a computer

Description automatically generated

In this popup two fields are “Source System Name” and “Description”.

After filling in these input fields and when we clicked on save button these both records will show on the main screen.A screenshot of a computer

Description automatically generated with medium confidence

In below screenshot we can see our source system is appearing

A screenshot of a computer

Description automatically generated with medium confidence**Edit**

A screenshot of a computer

Description automatically generated with low confidence

Using this edit button we can edit the existing source system. Once we click on this button one popup will come with existing values.

A screenshot of a computer

Description automatically generated with medium confidence

Once we click on the save button change will appear on main screen.

**Delete**

Once we click on the delete button that row will be deleted, and it will not appear on the screen. If any Source system which is linked with contract that Source System, we can’t delete, it will delete it only when there is no contract associated with that Source system.

A screenshot of a computer

Description automatically generated with low confidence

**API calls Source System**

|  |  |  |
| --- | --- | --- |
| **GET**  **/administrations/source/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/source/search?page=1&limit=75** | **API for search-(payload not required)** |
| **POST**  **/administrations/source/saveSourceSystem** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/source/saveSourceSystem** | **API for saveSourceSystem**  **(payload required {"sourceSysId": null,"sourceSysName": "test614","sourceSysDesc": "test614"})** |
| **DELETE**  **/administrations/source/deleteSourceSystem** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/administrations/source/deleteSourceSystem?sourceSysId=44** | **API for deleteSourceSystem (payload required in queryParameter sourceSysId : <id> )** |

## Vendors

In this screen we have three columns namely:

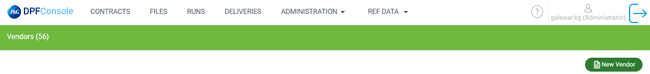
* Name
* Code
* Description

A screenshot of a computer

Description automatically generated with medium confidence

**New Vendor**

This button is there in Header.



Once we click on this button a popup will come

A screenshot of a computer

Description automatically generated with medium confidence

In this we have three fields “Vendor Name”,” Vendor Code”,” Description”. These all are mandatory fields. Once we fill in all details then on clicking on save button will save the data and a new row will appear on the main screen with the saved data

A screenshot of a computer

Description automatically generated with medium confidence

**Edit**

A screenshot of a computer

Description automatically generated

Using this edit button we can edit existing Vendors. Once we click on this button a popup will open with existing values.

A screenshot of a computer

Description automatically generated with medium confidence

We can modify the existing values and once we click on the save button change will appear on main screen.

**Delete**

Once we click on the delete button that particular row will be deleted, and it will not appear on the screen. If any vendor which is linked with contract that vendor we can’t delete, it will delete it only when there is no contract associated with that vendor.

A screenshot of a computer

Description automatically generated

**API call Vendors**

|  |  |  |
| --- | --- | --- |
| **GET**  **/vendor/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/vendor/search?page=1&limit=75** | **Api for search-(payload not required)** |
| **POST**  **/vendor/saveVendor** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/vendor/saveVendor** | **Api for save vendor**  **(payload required in request body :**  **{"vendrId": null,"vendrCode": "test621","vendrName": "test621","vendrDesc": "test621"})** |
| **DELETE**  **/vendor/deleteVendor** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/vendor/deleteVendor?id=34** | **API for delete Vendor**  **(payload required in queryParameter : Id:<id>)** |

## Integrated Contract

In this screen we have

* Data Tier
* Processing Type
* Integrated Contract Code
* Data Owner
* Status
* Category Name
* Type Code
* Atomic Contract Code
* Atomic Contract Name
* Country Name
* Promotion Indicator

A screenshot of a computer

Description automatically generated with medium confidence

There are two buttons in header **Discard Changes** and **Save Changes**

A screenshot of a computer

Description automatically generated with medium confidence

**Discard Changes**

By Default, this button is disabled and on toggling the “Promotion Indicator” switch this button will enable. Once we click on this button all changes done in promotion indicator column will revert.

A screenshot of a computer

Description automatically generated with medium confidence

**Save Changes**

By Default, this button is disabled and on toggling the “Promotion Indicator” switch this button will enable. Once we click on this button all changes done in promotion indicator column is saved and it will be updated in the Database.

A screenshot of a computer

Description automatically generated with medium confidence

**API for integrated Contract**

|  |  |  |
| --- | --- | --- |
| **GET**  **/dataPromotion/search** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/dataPromotion/search?page=1&limit=75** | **API for search-(payload not required)** |
| **PUT**  **/dataPromotion/update** | **https://cloudpanel-ui-api.azurewebsites.net/dpf/dataPromotion/update** | **API for savechanges**  **(payload required in request body :**  **entire row data)** |

# Authentication And Authorization:

Approach: Authentication and Authorization is implemented using Spring Security’s OAuth2 Server and React.js’s MSAL Client along with Azure Active Directory.

A picture containing text, screenshot, diagram, design

Description automatically generated

**Flow diagram - 1:**

DPF Console UI Flow (Detailed Flow of Part A of above Flow diagram - 1):

**A close-up of a computer screen

Description automatically generated**

Points 1.1, 1.2 & 1.3 of Part A from Flow diagram – 1 would be the part of below DPF Console UI flow.

Steps of Flow diagram – 1:

**Part A:**

* User logs into DPF Console Web App (<https://cloudpanel-ui.azurewebsites.net/>)
* Users will be redirected to P&G Microsoft Login page.
* After successful login, the user will be redirected to DPF Console dashboard.
* Tokens will be stored in the application session storage.

**Part B:**

Frontend application will send Authorization token in header to each backend request and the token which is being validated by Spring Security. If the request has a valid token, it will return a response. Else it will send https 401 Unauthorized.

## Authentication:

To Access the DPF console application. First user should get authenticated by providing valid credentials like user mail id and password.

### Steps of DPF Console UI Secured Login:

User logs into DPF Console Web App (https://cloudpanel-ui.azurewebsites.net/)

A screenshot of a computer

Description automatically generated

Users need to enter P&G credentials in P&G Microsoft Login page.

A screenshot of a computer

Description automatically generated

Users should complete the multi factor authentication with PING ID.

A screen shot of a computer

Description automatically generated

After successful login, the user will be redirected to DPF Console dashboard.

A screenshot of a computer

Description automatically generated

On the top right corner Username, user role and a Logout button are displayed.

A screenshot of a computer

Description automatically generated

If user enters wrong credentials, then user will stay on P&G Microsoft Login page along with error message.

Default P&G Account standards will be followed for User Login attempts.

A screenshot of a computer screen

Description automatically generated

### Authentication Error:

If the User is not assigned to the Azure Active directory DPF console application through user group in enterprise application of Active directory. User will not be able to login to the DPF console app. Below error screen will be displayed.

A screenshot of a computer

Description automatically generated

### Steps of DPF Console UI Logout:

When user clicked on Logout button, it will clear the stored user session from Application Session of the browser which contains user token details. After successful logout, Sign into your account screen will be displayed.

A blue and white text on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Authorization:

Authorization is implemented to give users permission to access some data or perform a particular action.

Role based Authorization is implemented for DPF console application.

### User Roles and AD groups:

Primarily user will have 4 roles Administrator, Support, Viewer and Edit.

Roles are mapped to Active Directory Group as mentioned in below table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Frontend Role** | **Role Alias** | **AD Group Non-Prod** | **AD Group Prod** |
| Viewer | viewerT1 | EDW-MMCONSOLE-T1-VIEW-NONPROD | EDW-MMCONSOLE-T1-VIEW |
| viewerT2 | EDW-MMCONSOLE-T2-VIEW-NONPROD | EDW-MMCONSOLE-T2-VIEW |
| viewerTHHP | EDW-MMCONSOLE-HHP-VIEW-NONPROD | EDW-MMCONSOLE-HHP-VIEW |
| Edit | editT1 | EDW-MMCONSOLE-T1-EDIT-NONPROD | EDW-MMCONSOLE-T1-EDIT |
| editT2 | EDW-MMCONSOLE-T2-EDIT-NONPROD | EDW-MMCONSOLE-T2-EDIT |
| editTHHP | EDW-MMCONSOLE-HHP-EDIT-NONPROD | EDW-MMCONSOLE-HHP-EDIT |
| Support | support | EDW-MMCONSOLE-SUPPORT-NONPROD | EDW-MMCONSOLE-SUPPORT |
| Administrator | admin | EDW-MMCONSOLE-ADMIN-NONPROD | EDW-MMCONSOLE-ADMIN |

**Note**: if user has support role and any of edit role like editT1 or editT2 or editTHHP. Front end Role will be **Edit/Support**.

**Administrator**:



**Support**:



**Edit**:

editTHHP:



editT1:



editT2:



**View**:

viewTHHP:



viewT1:



viewT2:



**Edit/Support:**



Below table describe the module wise access based on user roles:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Functionality | Type of User | | | | |
| Tier1 | Tier2 | HHP | Support | Administrator |
| Contracts | Yes | Yes | Yes | Yes | Yes |
| Files | No | Yes | Yes | Yes | Yes |
| Run | Yes | Yes | Yes | Yes | Yes |
| Delivery | Yes | Yes | No | Yes | Yes |
| Administration | No | No | No | Yes | Yes |
| Ref Data | Yes | No | No | Yes | Yes |
| Ref Data -> Hhp mapping | No | No | Yes | Yes | Yes |

**Note:** Few functionalities are disabled or enabled based on security scope parameters of the user group. Only **Administrator** can add, modify, or delete a Security scope (Please Refer Administration --> Security Scope Section)**.**

Actions in contracts, runs and Deliveries screen are accessible if contract matches with security scopes (compares data tier, data owner, category, region). View Users will have view only access and Edit Users will have New, modify, Remove Access.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

API calls Microsoft Graph:

|  |  |  |
| --- | --- | --- |
| **GET**  **/v1.0/users/{id}/memberOf/microsoft.graph.group** | **https://graph.microsoft.com/v1.0/users/{id}/memberOf/microsoft.graph.group?$count=true&$orderby=displayName&$select=id,displayName** | **API call to get all the groups that the user is a direct member of** |
| **GET**  **/v1.0/groups** | **https://graph.microsoft.com/v1.0/groups?$select=id,displayName&$count=true&$search=\"displayName:{groupName}** | **API call to check group exists in AD** |

Please find below sheet which contains module/sub module wise role access and respective groups.



### DB tables/views used for the authorization:

Below table & views are used to add or remove authorization for a user group over the User roles.

mmc\_scrty\_grp\_scope\_uprc

Table ***mmc\_scrty\_grp\_scope\_uprc*** will have the AD group name mapped with below properties:

Country/Region, Category, Contract, Data Owner, Data Tier.

mmc\_scrty\_grp\_scope\_vw

View ***mmc\_scrty\_grp\_scope\_vw*** will have the AD group name against aggregation of fields/properties (list of properties): Country/Region, Category, Contract, Data Owner, Data Tier.

mmc\_cntrt\_grp\_vw

View ***mmc\_cntrt\_grp\_vw*** will have mapping of AD group name and Contract.

## Azure Active Directory App Registration:

**Dev**: cloudpanel-app

**Prod**: cloudpanel-prod-app

App roles:

A screenshot of a computer

Description automatically generated

Assigned App Roles to User Groups in enterprise Application:

A screenshot of a computer

Description automatically generated

# Azure Integration

## Azure Key Vault Integration:

Azure Key Vault cloud service is used for securely storing secrets, such as database connection strings and Metadata API client secrets of DPF console application.

A computer screen shot of a diagram

Description automatically generated

Azure key vault flow

Steps Involved:

The backend application will send a request to azure active directory to get a token result for scopes and the token is authenticated.

Required properties values are fetched from Azure Key vault by passing the key to the property.

Connecting to Data base or Metadata API can be obtained through the fetched properties from the Key Vault

### DPF Console application Key vaults:

**Dev**: kvtpd1n1

**Prod**: kv-cloudpanel-prod

DEV:

**Resource group:** AZ-RG-CoreDataPlatform-TradePanel-Data-01

**Subscription:** PG-NA-External-Prod-02

**Vault URI:** <https://kvtpd1n1.vault.azure.net/>

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

PROD:

**Resource group:** AZ-RG-CoreDataPlatform-CloudPanel-Main-PROD

**Subscription:** PG-NA-External-Spoke-CDL-Prod-02

**Vault URI:** <https://kv-cloudpanel-prod.vault.azure.net/>

A screenshot of a computer

Description automatically generated

## Continuous integration and continuous deployment:

Continuous integration:

Below actions are performed as a part of continuous integration

clean and package

code coverage

sonar

push artifact (mvn deploy)

Continuous deployment:

Below actions are performed as a part of continuous deployment

download artifact.

deploy web app.

# Chain/graph structure

|  |  |  |  |
| --- | --- | --- | --- |
| Frequency | Tier1 | Tier2 | HHP |
| Weekly | x | x | NA |
| Monthly | x | x | NA |
| Bi-Monthly | x | x | NA |
| Quarterly | NA | o | NA |
| Custom | NA | o | NA |

Above table represents graph/process Vs Frequency corresponding to them. HHP fact values not spread by frequency.

## DPF\_Tier1\_Atomic TP Main Chain SFF

Initially when a contract is created for file provided by vendor/user, created contract obey vendor file pattern, contract code, file format, deliverable fact frequency based on TIME\_PERD\_CLASS\_CODE, measure multiplicators respectively. Graph/Process starts when a file dropped in IN folder naming convention matching with created contract file pattern. By processing the file, it fetches new data/ updating the old data based on requirement for product, market and fact followed by publishing.

### File Parsing

The Atomic TP Main chain SFF contains below mentioned file formats:

* SFF
* SFF2

The input/loaded file in IN Folder contains five different files as categorized product, market, time, measure, fact.

**Product File Parsing**

* This process starts with loading the file from the main file with matching condition as %PROD% and further this file will be materialized.
* For SFF and SFF2 the only difference in this file loader is, in SFF2 short is mapped with ATTR\_CODE\_LIST and long with EXTRN\_NAME. In SFF it is vice versa.

**Market File Parsing**

* This process starts with loading the file from the main file with matching condition as %MKT% and further this file will be materialized.

**Time File Parsing**

* This process starts with loading the file from the main file with matching condition as %PER% and further this file will be materialized.

**Run Logs Transformation**

* Map input time period ref data takes input as materialized time and checks the matching records from MM\_TIME\_PERD\_ID\_LKP, MM\_TIME\_PERD\_FDIM using left join based on conditions.
* Prepare data for Run Time Periods adds required parameters.
* Prepare data for Run Partitions adds required parameters.
* Publish the processed data for run time periods using insert.
* Publish the processed data for the run partitions log using insert.

**Measure file Parsing**

* This process starts with loading the file from the main file with matching condition as %FCT% and further this file will be materialized

**Fact file Parsing**

* This process starts with loading the file from the main file with matching condition as %FCT% and further this file will be materialized.
* In initially materialize file will format as per our requirements. Here we will have MKT\_TAG, PROD\_TAG, PER\_TAG columns
* in final materialize the columns will map with mkt extrn code, prod extrn code, time extrn code, and remaining all as fact\_amt\_1 so on.

**Atomic Product Transformation**

* Copy MM\_PROD\_SDIM, MM\_PROD\_XREF, MM\_PROD\_DIM for the processed contracts.
* Get Product attribute Names, column names: Input (materialized product) matching records (left outer join) on MM\_CATEG\_STRCT\_ATTR\_ASSOC\_VW with specific filter.
* Normalize Product Attribute codes to Columns: we are converting all the ATTR\_PHYS\_NAME to columns based on Case when statement.
* Get Product Attribute Value: we are performing left outer join on MM\_PROD\_CXREF (processed contract) and output of first multi joiner to get the attribute value.
* Normalize Product Attribute Values to Columns: we are converting all the ATT\_VAL to columns based on the case when statement.
* Recognize Product with “item name change”: we are updating the normalized output if any product with item name change and merge it to MM\_PROD\_CXREF.
* Match Product by “External code match”: we are updating the normalized output and match product by external code and merge it to MM\_PROD\_CXREF
* Match Product by Match “Attribute list Match”: we are updating the normalized output and match product by Attribute code list and merge it to MM\_PROD\_CXREF.
* Find New Product: We are updating the normalized output if any new product are found and then merge it to MM\_PROD\_CXREF
* For the New products PROD\_SKID will be mapped using Surrogate Key Lookup.
* Publish Product Data into MM\_PROD\_XREF: Before Publish we must load the MM\_PROD\_XREF (not processed) with semaphore option(exclusive) and merge it with MM\_PROD\_XREF (processed) table then publish the output data. After publishing release the semaphore using Semaphore operator operation.
* Assign products SKIDS for Input data: Mapping the Prod skid for the existing products.
* Find New and changed Rows for Product SDIM Data: Here we are checking null in prod skid and setting the Flag (I, U,E) based on the conditions and we filter out flag ! E i.e., not existing product in MM\_PROD\_SDIM
* Normalize Product Attributes Values: we are normalizing the Attribute value of MM\_PROD\_CSDIM
* Get Product Name and Description: We are extracting the Product Name Description using the function from Stored proc.
* Set Product Name and Description: We are adding prod name, prod desc columns by performing case statements on Prod level name column.
* Find new and Changed Products in DIM Table: Here we are checking null in prod skid and setting the Flag (I, U,E) based on the conditions and we filter out flag ! E i.e., not existing product MM\_PROD\_DIM

**Atomic TP Market Transformation**

* Assign Market SKID for TP Input Markets: Input as materialized market and reference table as MM\_MKT\_SKID\_LKP performing left outer join with reference to get matching records from reference based on condition.

### Nested graph: T1\_SFF Fact Standardization

* Convert Fact Input Columns to Numbers: using case statement we convert the input fact columns to numbers.
* Prepare fact data for DVM: checking nulls in the fact columns based on number of measures in the source measure file.
* Map input measures reference data: input as materialized measure and ref as MM\_MEASR\_ID\_LKP, checking for the matching records using left join.

### Nested graph:T1\_SFF Run Logs Loading

* Publish MM Market Run Log using insert mode: By performing left anti join the input data brings only the rows that does not have matching records from the reference(MM\_RUN\_MKT\_PLC).
* Publish MM Product Run Log using insert mode: By performing left anti join the input data brings only the rows that does not have matching records from the reference(MM\_RUN\_PROD\_PLC).

### Nested graph:T1\_SFF\_Atomic Measure Calculation

* Get Exchange rates: Based on Time\_perd\_id of time mapping and the EXCHANGE\_RATE\_FACT\_VM table perform left outer joiner to get Exchange rate.
* Standardize Measure names and units: mapping the fact amt columns to the correct measures from source measure file dynamically.
* Add currency exchange rates to fact data: to get matching records from both the tables based on time extrn code from mapped columns and in exchange rate with extrn code.
* Calculate currency Measures creates new columns by multiplying with corresponding exchange rates.
* Compute row-based Measures: Performing set of operations defined for each measure if the measures are existing in the mapped measures to the fact amt columns.
* Compute row-based Measures 2nd calculations: : Performing set of operations defined for each measure if the measures are existing in the mapped measures to the fact amt columns.

### Nested graph: T1\_SFF Fact Dimension Skid

* Assign dimension skid for fact: after computing row-based measure, we will assign prod skid and mkt skid to the input data frame.
* Create Fact Image: selects the required columns for the processed data having related frequency.

### Nested graph: T1\_SFF Fact Image Derivation

* Materialize Fact Image: In simple terms creates a data frame with frequency like MTH, WK,BIMTH of the processed fact data.

### Nested graph: T1\_SFF\_Measure\_log\_Loading

* Prepare Mm Measure Run Log: By performing the joiner on MM\_MEASR\_LKP and FCT\_CRBM2 check for the measure and prepare the data for publishing as log
* Publish Mm Measure Run Log: by performing left anti join the input data brings only the rows that does not have matching records from the reference (MM\_MEASR\_PLC)

### Nested graph: T1\_SFF Fact Copy

* Select Distinct Time Periods from Fact Image: step is to select the distinct object from the required frequency of the contract (MTH,BIMTH,WK).
* Select Min and Max Time Period End Dates: Using Aggregator condition selecting Min and Max Time Period End Dates.
* List All Expected Time Periods: selecting MM\_TIME\_PERD\_FDIM (ref) on column Time period type code, input data selects the matching records based on time period end date, for min time period end date it should be MIN\_MM\_TIME\_PERD\_END\_DATE from processed is less than or equal to reference TIME\_PERD\_END\_DATE and for max time period end date it should be MAX\_MM\_TIME\_PERD\_END\_DATE for processed is greater than or equal to reference TIME\_PERD\_END\_DATE.

### Nested graph:T1\_SFF Business Data Quality

* Business data quality measures all the business parameters that to be considered starts with loading fact schema and column complementing with the processed fact image data followed by partitioning of columns cntrt\_id, srce\_sys\_id, prod\_partition\_code, mm\_time\_perd\_end\_date. Tables MM\_DVM\_CNTRT\_STRCT\_PRC2\_VW, MM\_DVM\_CNTRT\_STRCT\_LVL\_PRC2\_VW are loaded from PostgreSQL and filtered by cntrt\_id and tables MM\_DVM\_RUN\_STRCT\_PLC, MM\_DVM\_RUN\_STRCT\_LVL\_PLC loaded from blob storage and perform left anti to fetch new records for processed cntrt\_id followed by column complementing and publishing.
* For the loaded table MM\_DVM\_CNTRT\_STRCT\_LVL\_PRC2\_VW we derives two filter conditions one based on cntrt\_id, another as cntrt\_id, top\_lvl\_ind as data frames, from product derivation df\_prod\_nav, df\_mm\_prod\_cxref to perform multi joiner for common columns using inner join defined as df\_dqm\_prod followed by column expression transformation with case conditions involved. For the same df\_dqm\_prod duplication involved to fetch parnt\_prod\_skid. using left outer join. In the respective data frame by applying nvl on columns parnt\_prod\_skid and prod\_skid using CET. This processed data will be published into MM\_TP\_VALDN\_PROD\_DIM.
* Table MM\_CNTRT\_TIER\_EXTND\_VW is loaded from PostgreSQL with filter condition on cntrt\_id, Run status view is created in generic operator followed by filter condition on cntrt\_id, time\_perd\_class\_code, process status. Table MM\_TP\_MTH\_FCT loaded from blob filtered by srce\_sys\_id, prod\_prttn\_code, fact\_type\_code. For fact data CET is applied on only run\_id,mkt\_skid, mm\_time\_perd\_end\_date, prod\_prttn\_code, cntrt\_id, srce\_sys\_id and distinct by srce\_sys\_id and from processes fact data also we fetch same columns with distinct merge together and forms a list(df\_all\_fct\_mkt\_list).
* For the list using inner join with the view(run status) created followed by distinct and aggregator to calculate max run\_id, cnt and publish the data into TIME\_PERD\_CNT. Using case statement in SQL syntax operator we find new\_time\_perd\_ind, using multi joiner
* Finding matching columns and common columns from the respective data frames for time\_perd\_end is “Y” and “N” with respective multi joiners and merge the data frames after column complementing(df\_dqm\_fct). Table MM\_TIME\_PERD\_FDIM\_VW loaded from blob, by using multi joiner left outer fetch the common values from respective data frames (df\_dqm\_calc\_measr ).
* From processed data applying CET to get data frames for df\_dqm\_calc\_measr, df\_dqm\_calc\_index, df\_dqm\_per\_publ.

### Nested graph: T1\_SFF\_Atomic Product Publication

* Publish processed product data into MM\_PROD\_SDIM: In this process of publishing processed data in product derivation with reference as MM\_PROD\_SDIM table, having two conditions like if the processed data has matching records, it updates the MM\_PROD\_SDIM or else if it found new records it will insert those values.
* Publish processed product data into MM\_PROD\_DIM: In this process of publishing processed data in product derivation with reference as MM\_PROD\_DIM table, having two conditions like if the processed data has matching records, it updates the dim or else if it found new records it will insert those values.

### Nested graph: T1\_SFF\_Atomic\_Fact\_Publication

* Publish Fact Data: Based on the frequency of the contract (MTH, BIMTH, WK) Publishing in the data into MM\_TP\_<<TIME\_PERD\_CLASS\_CODE>>\_FCT.

**Flow Diagram:**



## DPF\_Tier1\_Atomic TP Main Chain

Initially when a contract is created for file provided by vendor/user, created contract obey vendor file pattern, contract code, file format, deliverable fact frequency based on TIME\_PERD\_CLASS\_CODE, measure multiplicators respectively. Graph/Process starts when a file dropped in IN folder naming convention matching with created contract file pattern. By processing the file, it fetches new data/ updating the old data based on requirement for product, market and fact followed by publishing.

### File Parsing

The Atomic TP Main chain contains below mentioned file formats:

* FFS
* FFS2
* Tape2/Tape3

**FFS/FFS2**:

The structure will populate the 4 axis of the databases; it means PROD, AREA, TIME and MEAS.

The first column identifies what is the dimension that the line will impact:

|  |  |
| --- | --- |
| G | GEOG |
| L | PROD |
| R | TIME |
| O | MEAS |

The Data (Fact):

The structure of the line is the following:

col 1 w 1: U

col 2 w 9: AREA code

col 11 w 9: PROD code

col 20 w 9: TIME code

col 53 w 10: first measure

col 63 w 10: second measure

Etc.

The order of the measure is the one found in the structure part of the file. It means that the first measure in the data part (col 29 w 10) is the first measure in the structure part of the file.

**Tape2/Tape3**:

**Area (Market) dedicated preprocessor**:

Print first 6 char of every line and check for line ending with 02

Ex: with Column ("ccc",col ("value")[7:2]).filter("ccc=02")

**Time dedicated preprocessor**

Print first 6 char of every line and check for line ending with 01

Ex: with Column ("ccc",col ("value")[7:2]).filter("ccc=01")

**Product dedicated preprocessor**

Print first 6 char of every line and check for line ending with 06

Ex: with Column ("ccc",col ("value")[7:2]).filter("ccc=06")

**Measure dedicated preprocessor**

Print first 6 char of every line and check for line ending with 04

Ex: with Column ("ccc",col ("value")[7:2]).filter("ccc=04")

**Facts dedicated preprocessor**

Line having first 65 char without any spaces.

### Nested graph: T1\_Product Derivation

* Copy MM\_PROD\_SDIM, MM\_PROD\_XREF, MM\_PROD\_DIM for the processed contracts.
* Get Product attribute Names, column names: Input (materialized product) matching records (left outer join) on MM\_CATEG\_STRCT\_ATTR\_ASSOC\_VW with specific filter.
* Normalize Product Attribute codes to Columns: we are converting all the ATTR\_PHYS\_NAME to columns based on Case when statement.
* Get Product Attribute Value: we are performing left outer join on MM\_PROD\_CXREF (processed contract) and joined values from multi join operator to get the attribute value
* Normalize Product Attribute Values to Columns: we are converting all the ATT\_VAL to columns based on the case when statement
* Recognize Product with “item name change”: we are updating the normalized output if any product with item name change and merge it to MM\_PROD\_CXREF.
* Match Product by “External code match”: we are updating the normalized output and match product by external code and merge it to MM\_PROD\_CXREF
* Match Product by Match “Attribute list Match”: we are updating the normalized output and match product by Attribute code list and merge it to MM\_PROD\_CXREF.
* Find New Product: We are updating the normalized output if any new product are found and then merge it to MM\_PROD\_CXREF
* For the New products PROD\_SKID will be mapped using Surrogate Key Lookup.
* Publish Product Data into MM\_PROD\_XREF: Before Publish we must load the MM\_PROD\_XREF (not processed) with semaphore option(exclusive) and merge it with MM\_PROD\_XREF (processed) table then publish the output data. After publishing release the semaphore using Semaphore operator operation.
* Assign products SKIDS for Input data: Mapping the Prod skid for the existing products.
* Find New and changed Rows for Product SDIM Data: Here we are checking null in prod skid and setting the Flag (I, U, E) based on the conditions and we filter out flag! E i.e., not existing product in MM\_PROD\_SDIM
* Normalize Product Attributes Values: we are normalizing the Attribute value of MM\_PROD\_CSDIM
* Get Product Name and Description: We are extracting the Product Name Description using the function from Stored proc.
* Set Product Name and Description: We are adding prod name, prod desc columns by performing case statements on Prod level name column.
* Find new and Changed Products in DIM Table: Here we are checking null in prod skid and setting the Flag (I,U,E) based on the conditions and we filter out flag ! E i.e., not existing product MM\_PROD\_DIM

### Nested graph: T1\_Market Derivation

* Assign Market SKID for TP Input Markets: performing join on MM\_SKID\_MKT\_LKP Table we are mapping mkt skid to inputs from source market file.

### Nested graph: T1\_Fact\_Standardization\_v1

* Convert Fact Input Columns to Numbers: we are checking for the NA and trimming spaces for all the fact columns from the source fact file.
* Consolidate fact lines: checking nulls in the fact columns based on number of measures in the source measure file.
* Complete measures columns to 100 for validations: Based on number of measures obtained in source measure file will perform coalesce operation on fact amt columns and for the rest measure we are casting Null to number

### Nested graph: T1\_Run\_Log\_Trans

* Map input time Period Reference data: Map input time period ref data takes input as materialized time and checks the matching records from MM\_TIME\_PERD\_ID\_LKP, MM\_TIME\_PERD\_FDIM using left join based on conditions.
* Prepare data for Run Time Period: Prepare data for Run Time Periods adds required parameters.
* Map Input Measure Reference Data: input as materialized measure and ref as MM\_MEASR\_ID\_LKP, checking for the matching records using left outer join.
* Prepare data for Run Partitions: Prepare data for Run Partitions adds required parameters.

### Nested graph: T1\_Atomic Measure Calculations

* Get Exchange rates: Based on Time\_perd\_id of time mappings and the EXCHANGE\_RATE\_FACT\_VM table perform left outer joiner to get Exchange rate.
* Standardize Measure names and units: mapping the fact amt columns to the correct measures from source measure file dynamically.
* Add currency exchange rates to fact data: to get matching records from both the tables based on time extrn code from mapped columns and in exchange rate with extrn code
* Calculate currency Measures creates new columns by multiplying with corresponding exchange rates
* Compute row-based Measures: Performing set of operations defined for each measure if the measures are existing in the mapped measures to the fact amt columns
* Compute row-based Measures 2nd calculations: Performing set of operations defined for each measure if the measures are existing in the mapped measures to the fact amt columns

### Nested graph: T1\_Fact Dimension Skid Assigning

* Assign dimension skid for fact: after computing row-based measure, we will assign prod skid and mkt skid to the input data frame.
* Create Fact Image: selects the required columns for the processed data having related frequency.
* In simple terms creates a data frame with frequency like MTH, WK,BIMTH of the processed fact data.

### Nested graph: T1\_Measure\_Log

* Prepare Mm Measure Run Log: By performing the joiner on MM\_MEASR\_LKP and FCT\_CRBM2 check for the measure and prepare the data for publishing as log
* Publish Mm Measure Run Log

### Nested graph: T1 Fact Standard to Months

* Select Distinct Time Periods from Fact Image: step is to select the distinct object from the required frequency of the contract (MTH, BIMTH, WK).
* Select Min and Max Time Period End Dates: Using Aggregator selecting Min and Max Time Period End Dates.
* List All Expected Time Periods
* List All missing Time Periods

### Nested graph: T1\_Runs\_log\_Loading

* Publish MM Run Time Periods Log.
* Publish MM Run Partitions Log.
* Publish MM Market Run Log.
* Publish MM Product Run Log.

### Nested graph: T1\_Atomic\_Product\_Publication

* Publish processed product data into MM\_PROD\_SDIM: In this process of publishing processed data in product derivation with reference as MM\_PROD\_SDIM table, having two conditions like if the processed data has matching records, it updates the MM\_PROD\_SDIM or else if it found new records it will insert those values.
* Publish processed product data into MM\_PROD\_DIM: In this process of publishing processed data in product derivation with reference as MM\_PROD\_DIM table, having two conditions like if the processed data has matching records, it updates the dim or else if it found new records it will insert those values.

### Nested graph: T1\_Atomic\_Fact\_Publication

* Publish Fact Data: Based on the frequency of the contract (MTH, BIMTH, WK) Publishing in the data into MM\_TP\_<<TIME\_PERD\_CLASS\_CODE>>\_FCT.

**Flow Diagram:**

****

## DPF\_Tier1\_Integrated TP Main Chain

DPF\_Tier1\_Integrated TP Main Chain/graph will be available in tradepanel/tradepanel app with git branch - dpf2cdl\_dev.

Integrated graph validates for whole tier1 graphs i.e., Chain ID 5000,13900. It triggers after the preprocessing stage is done for respective contracts. It usually starts with updating delivery details for the contract based on process run key, cntrt\_id by updating all the run details in the PostgreSQL table MM\_DLVRY\_RUN\_LKP.

### Nested graph: T1 Integrated Product Transformation\_v1

Table MM\_PROD\_DIM2IDIM\_VW, MM\_PROD\_XREF loaded from blob storage, MM\_IGRTD\_LAYER\_CNTRT\_ASSOC\_PRC loaded from PostgreSQL with respective filters for each table.

Using multi join operator MM\_PROD\_DIM2IDIM\_VW acts as input, remaining two tables acts as reference, from MM\_PROD\_XREF using join it pulls data those are matching and from MM\_IGRTD\_LAYER\_CNTRT\_ASSOC\_PRC it pulls the common values user inner join, for this processed data deduplication operator is applied to remove duplicates.

Table MM\_PROD\_SKID\_LKP is loaded from blob and fetches new records from the deduplicated-processed data using left anti join followed by column complementing and partition using CET operator MM\_PROD\_SKID\_LKP(new records) are published.

The published data is loaded using file loader and refreshed using generic, multi joiner helps to assign integrated prod\_skid to deduplicated atomic products using inner join. After that map the atomic products to the integrated products.

### Nested graph: T1 Integrated Market Transformation\_v1

Tables MM\_MKT\_DIM, MM\_CNTRY\_REGN\_VW loaded from respective areas and joined together to get matching records from MM\_CNTRY\_REGN\_VW using left outer followed by renaming of mkt\_skid to atomic\_mkt\_skid.

### Nested graph:T1 Integrated TP Market Association Loading\_v1

Table MM\_MKT\_DIM loaded from blob and MM\_STRCT\_LVL\_LKP, MM\_STRCT\_LKP from PostgreSQL. Using generic we set the parent market skid for processed integrated markets.

### Nested graph: T1 Integrated TP Product Association Loading\_v1

Table MM\_PROD\_SKID\_LKP loaded and using product transformation add skid data we set the natural key. After that the keys are set for processed integrated products.

### Nested graph: T1 Integrated Fact Transformation\_v1

Tables MM\_TP\_BIMTH\_FCT, MM\_TP\_MTH\_FCT, MM\_TP\_WK\_FCT loaded and filtered by srce\_sys\_id, limited the measures by respective atomic facts.by performing CET operator we get distinct time\_periods for processed fact data.

For weekly/monthly/bimonthly fact data assign prod skid, market skid and remap atomic dimension skids to integrated skids for respective fact data followed by deduplication.

### Nested Graph: T1 Integrated Market Custom Aggregate Calculation\_v1

Table MM\_TOT\_CNTRY\_CNTRT\_ASSOC\_VW loaded from blob and perform inner join to Getting fact data for custom aggregators and group by some columns. recalculation is done after aggregating followed by materialization. MM\_TIME\_PERD\_ASSOC\_IGRTD\_VW from blob and standardized using CET operator.

Using generic standardize these custom aggregates to months. For the respective process run key in MM\_RUN\_PRTTN\_PLC we compute the measures based on aggregate measures with distinct values. Get total country loaded files for fact data using inner join and filter the processed data with condition.

### Nested Graph: T1 Integrated TP Market Publication\_v1

Tables MM\_MKT\_DIM loaded with respective filters and merged together followed by partitioning and publishing the market dimension for the respective process run key.

### Nested Graph: T1 Integrated Product IDIM Contract Loading\_v1

List Distinct Prod Skids From Transformed Fact Data using distinct operator and by inner join with processed fact data we fetch available products and casted the cntrt\_id as float using CET.

### Nested Graph: T1 Integrated Market IDIM Contract Loading\_v1

List Distinct mkt Skids From Transformed Fact Data using distinct operator and by inner join with processed fact data we fetch available markets and casted the cntrt\_id as float using CET.

### Nested graph: T1 Integrated Dimension Publication\_v1

Table MM\_PROD\_IDIM is loaded from blob and complemented with processed fact data, this data is merged to MM\_PROD\_IDIM having reference as same and published with partition on srce\_sys\_id. Similarly, MM\_PROD\_ASSOC, MM\_MKT\_ASSOC, MM\_PROD\_CNTRT\_IDIM, MM\_MKT\_CNTRT\_IDIM are followed the same way.

**Flow Diagram:**



## DPF\_Tier2\_USA Nielsen fact categ main chain

DPF\_Tier2\_USA Nielsen fact categ main chain /graph will be available in tradepanel/tradepanel app with git branch - dpf2cdl\_dev. Initially when a contract is created for file provided by vendor/user, created contract obey vendor file pattern, contract code, file format, deliverable fact frequency based on TIME\_PERD\_CLASS\_CODE, measure multiplicators respectively.

Graph/Process starts when a file dropped in WORK folder naming convention matching with created contract file pattern. By processing the file, it fetches new data/ updating the old data based on requirement for product, market and fact followed by publishing.

### Nested graph: CDL\_TP\_USA Nielsen Fact-Load Ref Tables

Initially in graph we load all the required tables as per requirement in a nested graph named as CDL\_TP\_USA Nielsen Fact-Load Ref Tables. In this nested graph at starting, we create a dummy schema as data frame to be used in further steps because without a data frame we can’t initiate the process.

As part of loading of files, MM\_PROD\_PRTTN\_XREF from Postgres using generic operator, MM\_MKT\_DIM\_VW from blob storage with srce\_sys\_id partition. MM\_PROD\_DIM\_VW from blob storage with srce\_sys\_id partition. connect\_mm\_usn\_prod\_dim\_vw view is created from MM\_PROD\_DIM\_VW.

MM\_PROD\_PRTTN\_XREF loaded from Postgres will be mapped with database column names in generic operator. This process is done to avoid mismatching of column names.

### Nested graph: DPF2CDL\_TradePanel\_MFT

The whole process will initiate if file is drop by a user in work folder, to make sure everything is good with file pattern and unzipping of file, using this generic operator in nested graph it will unzip the provided file by using functions and if/else condition.

**DPF\_Tier2\_USA Nielsen fact categ main**

Coming to main graph, importing of list columns, vendor file patten. Step file pattern common and added to the mapped columns with cntrt\_id and mapping will be done for all measre columns with fact type code.

The mapped columns from database and the fact columns produced are multiplied with multiplicators for the fact columns using table MM\_MEASR\_LKP\_VW and updates the delivery details.

In column expression transformation from the generated extrn\_prod\_id we creates a new column pg\_categ\_txt having same values as extrn\_prod\_id.

In standardizing the data for mm\_time\_perd\_end\_date we are changing date format and adding of columns that are mentioned in graph config to be included in further transformations and renaming some of the columns.

In partition of columns, srce\_sys\_id, cntrt\_id, mm\_time\_perd\_end\_date as part\_srce\_sys\_id,part\_cntrt\_id, part\_mm\_time\_perd\_end\_date the partitions will help while publishing of data.

The retention date is calculated from the latest time period end date minus the retention period. This retention date is added as a column to the dataframe. The input is then filtered to take the data above this retention date.

We are assigning prod\_prttn\_code for the products from prod\_prttn\_xref table based on the pg\_categ\_txt.

To find the new records in distinct orphans comparing with df\_connect\_mm\_usn\_prod\_dim\_vw using left anti join. For the new records we assign prod skid as numeric datatype using surrogate lookup operator.

For prod dim schema creation is done to column complement, it is used as reference for the processed data to check the datatypes and columns, if columns do not exist in processed data, then CC operator will add the column null as value. After that the two processed data frames merge using merger operator with ref as connect\_mm\_usn\_prod\_dim\_vw and output as product dimension

The output is partitioned based on srce sys id. The next steps are implemented as part of semaphore, publishes data in prod\_dim vw. After publishing it releases the semaphore for the path.

### Nested graph: CDL\_TP\_USA Nielsen Fact-Load Dimensions

In main chain we perform distinct for market dimensions, for that data we are preparing some required columns like srce\_sys\_id, cntrt\_id in column expression transformation. This processed data and ref (MM\_MKT\_DIM) will have a left outer to get matching records from reference table. And after that it will have a left anti join to find new records in processed data.

Here mkt\_skid comes into picture for the new records. These mkt\_skids are assigned in surrogate lookup operator. Processed data column complements with reference data mm\_mkt\_dim.

In main chain we partition the complemented data with column srce\_sys\_id. After these we implement semaphore and publishes market data and releases semaphores for the path.

### Nested graph: CDL\_TP\_USA Nielsen Fact-WK Facts

For the stagging of tp\_wk\_fct\_stg implementation of partition based on prod\_prttn\_code this will represent fact data. In generic prod data is loaded as per requirement.

The multijoin is the key operator which adds the skids from prod, market and time\_perd\_id from time and all the fact columns from the stagging.

Creation of fact schema is done using create schema operator by using this data the stagging data, will be column complemented with data frame name and in another generic it finds out the max (MM\_TIME\_PERD\_END\_DATE).

In down streams fact schema from Postgres is loaded and this data will act as reference to through this column complement we get the final fact data.

In main chain the semaphore implementation will start by loading the fact data as reference. For this reference data from input, we find out the retention period because to match the retention period for the processed data and column compliments with fact schema as reference.

Publishes the fact data by merging processed data and reference data by partitioning and releases the semaphore.

After the whole process is done the file moved from work folder to archive folder in storage account.

**Flow Diagram:**

****

## DPF\_Tier2\_USA Nielsen fact item main chain

DPF\_Tier2\_USA Nielsen fact item main chain /graph will be available in tradepanel/tradepanel app with git branch - dpf2cdl\_dev. Initially when a contract is created for file provided by vendor/user, created contract obey vendor file pattern, contract code, file format, deliverable fact frequency based on TIME\_PERD\_CLASS\_CODE, measure multiplicators respectively.

Graph/Process starts when a file dropped in WORK folder naming convention matching with created contract file pattern. By processing the file, it fetches new data/ updating the old data based on requirement for product, market and fact followed by publishing.

### Nested graph: CDL\_TP\_USA Nielsen Fact-Load Ref Tables

Initially in graph we load all the required tables as per requirement in a nested graph named as CDL\_TP\_USA Nielsen Fact-Load Ref Tables. In this nested graph at starting, we create a dummy schema as data frame to be used in further steps because without a data frame we can’t initiate the process.

As part of loading of files, MM\_PROD\_PRTTN\_XREF from Postgres using generic operator, MM\_MKT\_DIM\_VW from blob storage with srce\_sys\_id partition. MM\_PROD\_DIM\_VW from blob storage with srce\_sys\_id partition. connect\_mm\_usn\_prod\_dim\_vw view is created from MM\_PROD\_DIM\_VW

MM\_PROD\_PRTTN\_XREF loaded from Postgres will be mapped with database column names in generic operator. This process is done to avoid mismatching of column names.

### Nested graph: DPF2CDL\_TradePanel\_MFT

The whole process will initiate if file is drop by a user in work folder, to make sure everything is good with file pattern and unzipping of file, using this generic operator in nested graph it will unzip the provided file by using functions and if/else condition.

**DPF\_Tier2\_USA Nielsen fact item main**

Coming to main graph, importing of list columns, vendor file patten. Step file pattern common and added to the mapped columns with cntrt\_id and mapping will be done for all measr columns with fact type code.

The mapped columns from database and the fact columns produced are multiplied with multiplicators for the fact columns using table MM\_MEASR\_LKP\_VW and updates the delivery details.

In standardizing the data for mm\_time\_perd\_end\_date we are changing date format and adding of columns that are mentioned in graph config to be included in further transformations and renaming some of the columns.

In partition of columns, srce\_sys\_id, cntrt\_id, mm\_time\_perd\_end\_date as part\_srce\_sys\_id,part\_cntrt\_id, part\_mm\_time\_perd\_end\_date the partitions will help while publishing of data.

The retention date is calculated from the latest time period end date minus the retention period. This retention date is added as a column to the data frame. The input is then filtered to take the data above this retention date.

We are assigning prod\_prttn\_code for the products from prod\_prttn\_xref table based on the pg\_categ\_txt.

To find the new records in distinct orphans comparing with connect\_mm\_usn\_prod\_dim\_vw using left anti join. For the new records we assign prod skid as numeric datatype using surrogate lookup operator.

For prod dim schema creation is done to column complement, it is used as reference for the processed data to check the datatypes and columns, if columns do not exist in processed data, then CC operator will add the column null as value. After that the two processed data frames merge using merger operator with ref as connect\_mm\_usn\_prod\_dim\_vw and output as product dimension

The output is partitioned based on srce sys id. The next steps are implemented as part of semaphore, publishes data in prod\_dim vw. After publishing it releases the semaphore for the path.

### Nested graph: CDL\_TP\_USA Nielsen Fact-Load Dimensions

In main chain we perform distinct for market dimensions, for that data we are preparing some required columns like srce\_sys\_id, cntrt\_id in column expression transformation. This processed data and ref (MM\_MKT\_DIM) will have a left outer to get matching records from reference table. And after that it will have a left anti join to find new records in processed data.

Here mkt skid comes into picture for the new records. These mkt skids are assigned in surrogate lookup operator. Processed data column complements with reference data MM\_MKT\_DIM.

In main chain we partition the complemented data with column srce\_sys\_id. After these we implement semaphore and publishes market data and releases semaphores for the path.

### Nested graph: CDL\_TP\_USA Nielsen Fact-WK Facts

For the stagging of tp\_wk\_fct\_stg implementation of partition based on prod\_prttn\_code this will represent fact data. In generic prod data is loaded as per requirement.

The multijoin is the key operator which adds the skids from prod, market and time\_perd\_id from time and all the fact columns from the stagging.

Creation of fact schema is done using create schema operator by using this data(df\_mm\_tp\_wk\_fct\_schema), the stagging data will be column complemented with data frame name as and in another generic it finds out the max (MM\_TIME\_PERD\_END\_DATE).

In down streams fact schema from Postgres is loaded and this data will act as reference through this column complement we get the final fact data.

In main chain the semaphore implementation will start by loading the fact data as reference. For this reference data from input, we find out the retention period because to match the retention period for the processed data and column compliments with fact schema as reference.

Publishes the fact data by merging processed data and reference data by partitioning and releases the semaphore.

After the whole process is done the file moved from work folder to archive folder in storage account.

**Flow Diagram:**

****

## DPF\_Tier2\_USA Nielsen main prod load chain

DPF\_Tier2\_USA Nielsen main prod load chain/graph will be available in tradepanel/tradepanel app with git branch - dpf2cdl\_dev. Initially when a contract is created for file provided by vendor/user, created contract obey vendor file pattern, contract code, file format, deliverable fact frequency based on TIME\_PERD\_CLASS\_CODE, measure multiplicators respectively.

Graph/Process starts when a file dropped in WORK folder naming convention matching with created contract file pattern. By processing the file, it fetches new data/ updating the old data based on requirement for product followed by publishing.

### Nested graph: CP\_T2\_import\_MM\_PROD\_PRTTN\_XREF

Nested graph contains the loading process of table- mm\_prod\_prttn\_xref through Postgres using generic operator, having filter on column srce\_sys\_id dynamically.

### Nested graph:CP\_create\_CONNECT\_MM\_PROD\_DIM\_VW

Graph contains file loading of both tables MM\_PROD\_DIM\_VW, MM\_PROD\_SDIM\_VW and lower column names of respective tables. later we are doing a left join operation to extract the matching columns from MM\_PROD\_SDIM\_VW with reference as same.

In column expression transformation creation of extrn\_prod\_id using a case statement by concatenation of some columns from the view created by join condition. filtering the data by srce\_sys\_id manually.

### Nested graph: DPF2CDL\_TradePanel\_MFT

The whole process will initiate if file is drop by a user in work folder, to make sure everything is good with file pattern and unzipping of file, using this generic operator in nested graph it will unzip the provided file by using functions and if/else condition.

Coming to main graph column mappings is done by reading the columns from Postgres adwgp\_mm\_consol.DPF\_COL\_ASIGN\_VW. unzipping the raw files doing mapping and sending data (run\_id, cntrt\_id, row\_count, file\_name, common\_file\_name) to PostgreSQL table (MM\_DLVRY\_RUN\_LKP).

Input schema is created for column complementation for mapped raw file input, for the complemented data column last\_sellg\_txt date format is modified by using CET (column expression transformation operator), then for the data deduplication operator is applied by partition pg\_categ\_txt, pg\_super\_categ\_txt,upc\_txt columns and ordered it by last\_sellg\_txt column. Filter the data by pg\_categ\_txt is NOT Null.

Creation of schema is made for aggregation as dummy. This schema is used as reference for the input data and aggregated data formed by input. In grouping of product data using aggregator operator, input data is group by pg\_categ\_txt, pg\_super\_categ\_txt.

For the CC (column complementor) data frames applying of merger operator is done to merge both data using reference as aggregated data. For the merged data applying standardization by including important columns run\_id, cntrt\_id, srce\_sys\_id, extrn\_prod\_id.

The data mm\_prod\_prttn\_xref which is extracted from PostgreSQL is converted to lower case to match the case condition(lower) with the input data. Same condition is applied to the df\_CONNECT\_MM\_PROD\_DIM\_VW.

For the processed data we apply left join with mm\_prod\_prttn\_xref to get the column prod\_prttn\_code from MM\_PROD\_PRTTN\_XREF and next apply left join with df\_CONNECT\_MM\_PROD\_DIM\_VW to get prod skid. For the new records apply prod skid using surrogate key lookup.

Data now has new records for this data frame we apply CC to complement with CONNECT\_MM\_PROD\_DIM\_VW. From CONNECT\_MM\_PROD\_DIM\_VW we apply left anti join on processed data to get new records from processed data and output named as respectively.

For respective data apply CC with CONNECT\_MM\_PROD\_DIM\_VW and merge data with processed dataframe as reference and names the data frame as product dimension

Creating schema for MM\_PROD\_DIM to complement with our processed data and after that adding partition by srce\_sys\_id before publishing, then the input file in work folder will move to archive.

**Flow Diagram:**



# HHP-Preprocessing

HHP chain/graph will be available in tradepanel/tradepanel app with git branch - dpf2cdl\_dev. Initially when a contract is created for file provided by vendor/user, created contract obey vendor file pattern, contract code, file format, deliverable fact frequency based on TIME\_PERD\_CLASS\_CODE, measure multiplicators respectively.

Graph/Process starts when a file dropped in WORK folder naming convention matching with created contract file pattern. By processing the file, it fetches new data/ updating the old data based on requirement for product followed by publishing.

DPF\_COL\_ASIGN\_VW is loaded from PostgreSQL to map the columns. unzipping the raw files doing mapping and sending data (run\_id,cntrt\_id,row\_cnt,file\_name, common\_file\_name) to PostgreSQL table( MM\_DLVRY\_RUN\_LKP). The data is filtered followed by case conditions and casted some columns based on respective conditions. Trimming of columns, coalesce and concatenate of columns done in generic preprocessing.

Filtered the processed data on fact columns. Adding of DVM columns to the processed data by using ‘if‘ condition, if DVM columns are not present in processed data. Filter by column time\_perd\_type\_code, casted time\_perd\_end\_date as string and distinct of data will be done in a sequence. MM\_TIME\_MAP\_LKP\_VW table loaded from blob and have a left anti(brings in only rows from the left table that don't have any matching rows from the right table) and publishes into MM\_TIME\_MAP\_LKP\_VW.

MM\_TIME\_MAP\_RESLT\_VW is loaded from blob and takes matching records from the processed input data using left outer and makes it as final output. For this time mapping mail sender data, conditional stop is applied to check the processed file condition using conditional stop operator.

**Note:** case conditions and casting, filtering of columns may vary depending on chains in preprocessing.

## Nested graph: HHP\_AtomicProduct

Atomic product starts with applying distinct operation on data which is transformed in atomic preprocessing at level 3. For this staging data deduplicated on primary column extrn\_prod\_id.

Deduplicated data further joins with table MM\_PROD\_SDIM and picks the records from it, on column extrn\_prod\_id. For the fetched new records where prod\_skid is null count is greater than zero. A table HHP\_PROD\_SKID created from that table; we get parent-child relation of data. For the processed data, column complemented with MM\_PROD\_SDIM followed by merger and partition by primary columns, data will be published into tables MM\_PROD\_SDIM and MM\_PROD\_DIM.

## Nested graph: HHP\_AtomicMarket

Atomic Market starts with applying distinct operation on data, which is transformed in atomic preprocessing at level 3, Stagging data further joins with table MM\_MKT\_SDIM and picks the matching records from it, on column extrn\_mkt\_id. For the fetched new records mkt\_skid is assigned. This data is further fetching new matching records from table GEO\_HIER\_DIM\_VW using left outer based on column iso\_cntry\_code. The processed data published into MM\_MKT\_SDIM, MM\_MKT\_DIM.

## Nested graph: HHP\_AtomicBuyerGroup

Atomic Market starts with applying distinct operation on data, which is transformed in atomic preprocessing at level 3, The stagging data fetches matching records from table MM\_BUYR\_GRP\_DIM on column BUYR\_GRP\_EXTRN\_ID using left outer join. For the new fetched matching records BUYR\_GRP\_SKID is assigned followed by column complementing with MM\_BUYR\_GRP\_DIM and published into MM\_BUYR\_GRP\_DIM.

## Nested graph: HHP\_AtomicFact

Tables MM\_PROD\_SDIM, MM\_MKT\_SDIM, MM\_BUYR\_GRP\_DIM are loaded, time mapping data from preprocessing as inputs to fact, The data is distinct followed by filter. The data is stored as log in MM\_RUN\_PRTTN\_PLC using insert condition, processed by left anti to get new data followed by column complementing data is published into MM\_RUN\_PRTTN\_PLC.

Data of time mapping and preprocessing fact from preprocessing(main graph). Here pre fact act as input, the data takes matching data from tables MM\_PROD\_SDIM, MM\_MKT\_SDIM, MM\_BUYR\_GRP\_DIM, MM\_TIME\_MAP\_RESLT\_VW using left outer based on respective conditions and further takes matching data from table from PostgreSQL MM\_CNTRT\_MEASR\_MULTR\_LKP\_VW followed by a case condition which multiplies the measure columns called as fact multiplier.

For the multiplied fact data, data gets the matching values from table MM\_TIME\_PERD\_FDIM\_VW using left outer join followed by case condition on coalesce on columns from input(fact multiplier) ref table MM\_TIME\_PERD\_FDIM\_VW and time\_perd\_type\_code, mm\_time\_perd\_end\_date respectively.

MM\_HHP\_CUSTM\_ARCHV\_FCT is loaded from PostgreSQL and column complemented with the processed fact and followed by partition and published. MM\_HHP\_CUSTM\_ARCHV\_FCT is loaded and filtered. DPF\_RUN from PostgreSQL and filtered, joined together with inner join based on column process run key and grouped by mm\_time\_perd\_end\_date,buyr\_grp\_skid,prod\_skid,time\_perd\_type\_code columns using aggregator operator on process run key. This max run data and fact distinct data together using inner join to get similar columns.

The processed data and MM\_MKT\_DIM\_VW, time mapping records together have a left outer join to get matching records followed by filter condition. Then the processed data again have a left outer to get matching records from MM\_TIME\_PERD\_FDIM. MM\_HHP\_CUSTM\_ARCHV\_FCT is loaded, and column complemented with the processed fact and followed by partition and published.

## Nested graph: HHP\_Intgtd\_Layer\_Publish

Integrated layer triggers after completion of main preprocessing for respective process run key of HHP contract.

MM\_PROD\_DIM,MM\_HHP\_PROD\_SLVR\_DIM\_VW,MM\_HHP\_SLVR\_BUYR\_GRP\_DIM\_VW,MM\_HHP\_DATA\_CNTXT\_RESLT\_VW,MM\_HHP\_THREE\_EDWM\_MAP\_VW,MM\_PROD\_IDIM, MM\_PROD\_SKID\_LKP tables are loaded as part of the integrated layer processing.

To start with product IDIM publishing from MM\_PROD\_SKID\_LKP table have to delete some data based on conditions on individual tables by doing joins, filters and column expression transformation, then the processed data for MM\_PROD\_SKID\_LKP acts as a reference table to products, to get matching records from it with a case statement. From the output removing of duplicate records are removed using deduplication operator. Then de duplicated records again checks the matching records with MM\_PROD\_SKID\_LKP, data acts as atmc\_itgrtd\_lkp.

Table MM\_HHP\_PROD\_SLVR\_DIM\_VW, MM\_HHP\_THREE\_EDWM\_MAP\_VW with filter condition, MM\_HHP\_DATA\_CNTXT\_RESLT\_VW and processed atmc\_itgrtd\_lkp data, together process data to get similar records and matching records from respective tables and this data act as hhp\_edwm\_prods followed by distinct of the data with filter act as exact\_prod\_trans and published data into MM\_PROD\_IDIM.

hhp\_edwm\_prods act as input and de-duplicated as reference, input brings out matching records from reference to get new child-parent relation followed by case condition. Then this processed data act as reference to table MM\_PROD\_DIM to get similar records using inner join act as decoded data, followed by distinct operator. The transformed data is merged into MM\_PROD\_DIM with column complementing data. Child- parent relation data is published into MM\_PROD\_SKID\_LKP similar to MM\_PROD\_DIM.

**Market IDIM refresh**

For the respective process run key table MM\_MKT\_DIM is loaded and filtered with srce\_sys\_id, column complemented with IDIM as reference and merged data into MM\_MKT\_IDIM using merge operator

**Buyer Group IDIM refresh**

For the respective process run key table MM\_HHP\_SLVR\_BUYR\_GRP\_DIM\_VW is loaded and filtered with srce\_sys\_id, column complemented with IDIM as reference and merged data into MM\_BUYR\_GRP\_IDIM using merge operator

**HHP Fact Derivation**

Table MM\_EDWM\_MAPNG\_HIST\_MM\_VW loaded from storage blob and filtered with table DPF\_RUN which is loaded from PostgreSQL based on condition where chng\_time\_stamp in input table is greater than max(start\_time\_stamp) from DPF\_RUN. This processed data act as input table and MM\_CNTRT\_CNTRY\_ASSOC as lookup table, checks data in MM\_CNTRT\_LKP and gets similar records from it using inner join. Acts as edwm\_time\_pre. Using this data for looking impacted contracts in the mapping’s changes for table MM\_HHP\_CUSTM\_ARCHV\_FCT by checking data in processed edwm\_time\_pre and DPF\_RUN(mm\_process\_run\_lkp\_vw), group by primary columns acts as time\_cntrt\_pre.

To filter out incorrectly mapped contracts inner join used between data edwm\_time\_pre as input, time\_cntrt\_pre as reference. Table MM\_HHP\_CUSTM\_FCT as input checks modified edwm products and buyer groups using reference data as edwm\_last\_chng, this data act as hhp\_edwm\_cntrt. Takes out only skids having the values.

Selecting new or modified contracts for the table MM\_CNTRT\_LKP using conditions like union and filters and this data act as new\_cntrt. To get atomic new fact data on process run key we use reference data’s new\_cntrt, edwm\_time\_cntrt, cntrt\_ddupato and takes out key fact data out,. Followed by case conditions and this data will get mkt skid using left join from the table MM\_MKT\_DIM and time related columns from the table MM\_TIME\_MAP\_RESLT\_VW act as FCT\_PRE\_P12M\_LKP.

Table MM\_TIME\_PERD\_FDIM is loaded and distinct, filtered act as time\_perd\_trans. This processed data act as reference to FCT\_PRE\_P12M\_LKP to get matching records from reference, distinct for the same and published as a log into MM\_RUN\_PRTTN\_PLC.

For fact data join is used to get similar skid records using inner join and left join, and integrated skid values from parent-child data using inner join followed by group by primary columns and partitioned the data and published into MM\_HHP\_CUSTM\_IFCT.

# Glossary

|  |  |
| --- | --- |
| **Terms** | **Abbreviations** |
| P&G | Proctor And Gamble |
| UI | User Interface |
| DB | Database |
| AD | Azure Active Directory |
| DPF | Data Processing Framework |
| DQ | Data Quality |
| PDF | portable document format |
| TP | Trade Panel |
| HHP | Household Product |
| API | Application Programming Interface |
| BI | Business Intelligence |
| VM | Virtual Machine |
| PROD | Production |
| SQL | Structured Query Language |
| ORM | Object Relational Mapping |
| T1 | Tier 1 |
| T2 | Tier 2 |
| T3 | Tier 3 |
| FFS | File Format Structure |
| EDWM | EDW Mapping Tool |

# ORM Mappings and DB Diagrams







# References

**DPF Console User Guide:**



**Authentication & Authorization Approach Document:**



**HLD Document:**



# FAQ

* + 1. How can user get the Id for newly created Source System Id from CloudPanel Console.
* Source System screen shows the source system screen Id
  + 1. How the newly created source system Id associated to the data provider key?
  + User goes into the system and creates the new source system. Then goes in to the rubik (MDM) and associates the newly created source system id to the data provider key.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

