MetLife

Navigating life together

GRID DataFrame Framework (GDF)

Building on solid foundations

Azure Databricks

Rich Kempinski

Investments Architecture

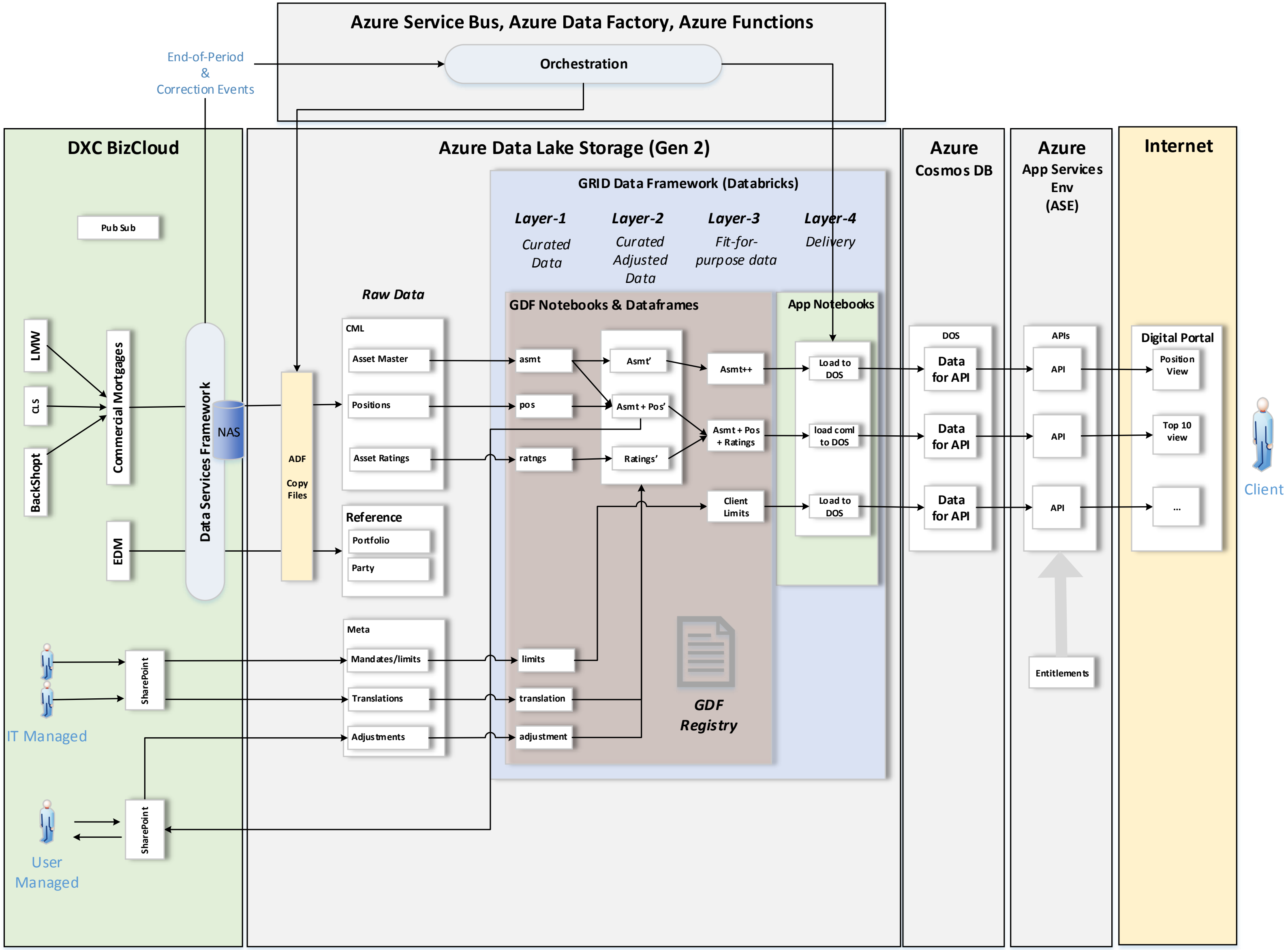
Oct 2019

Objectives

To provide an overview of

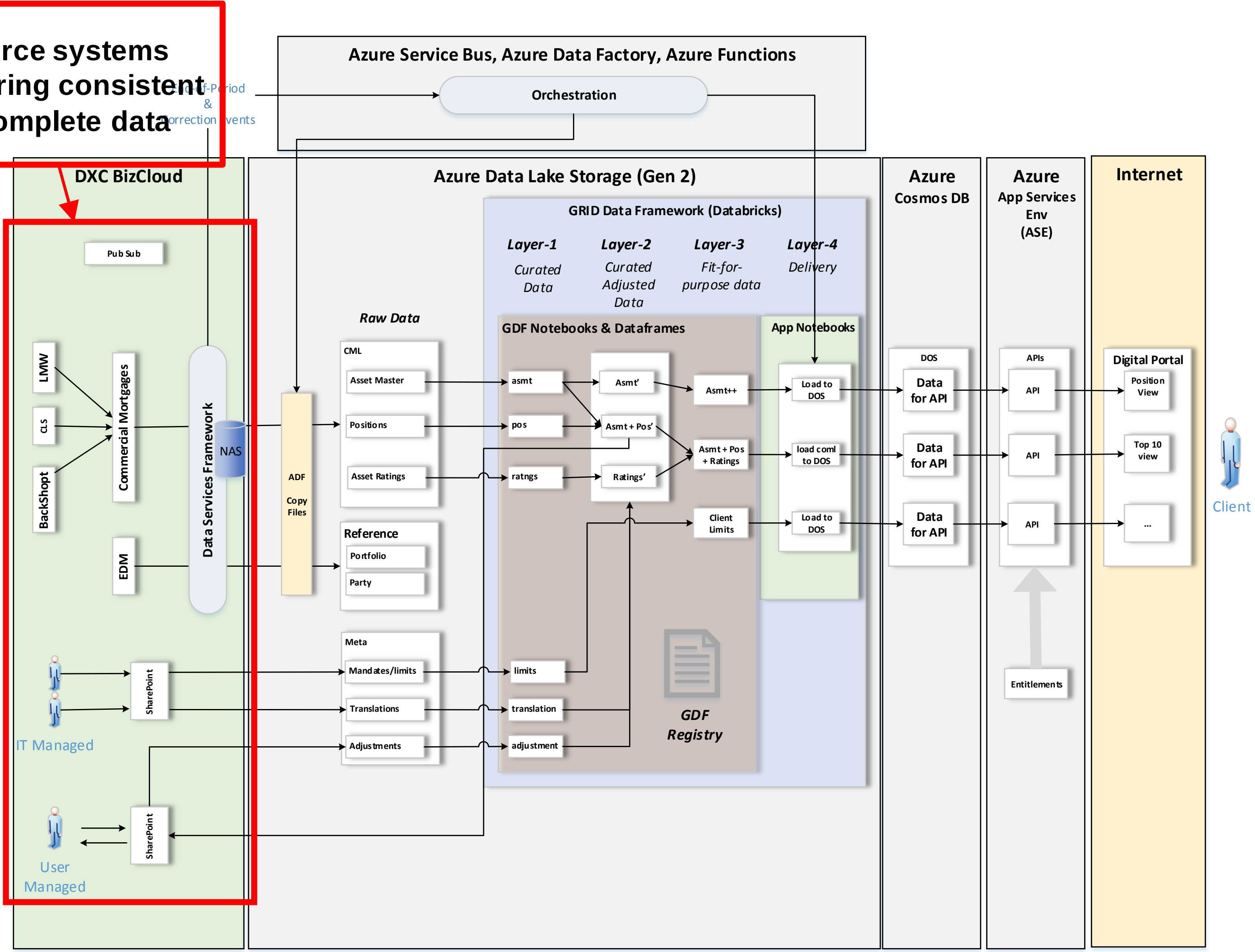
- End-to-end Data Architecture for Digital
- Azure Databricks
- GRID Dataframe Framework (GDF)

Overall Digital End-to-end Architecture



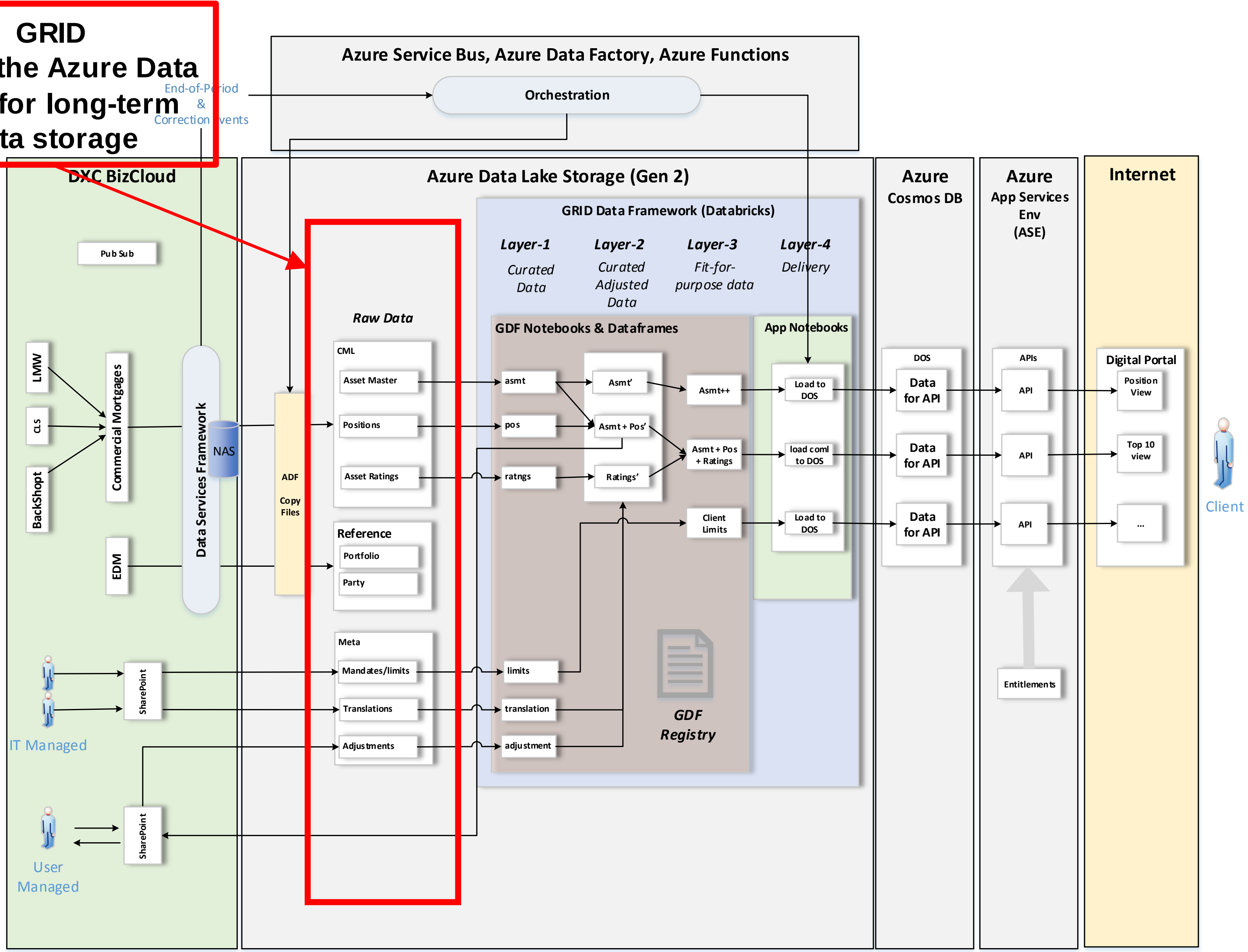
Overall Digital End-to-end Architecture

Source systems delivering consistent & complete data

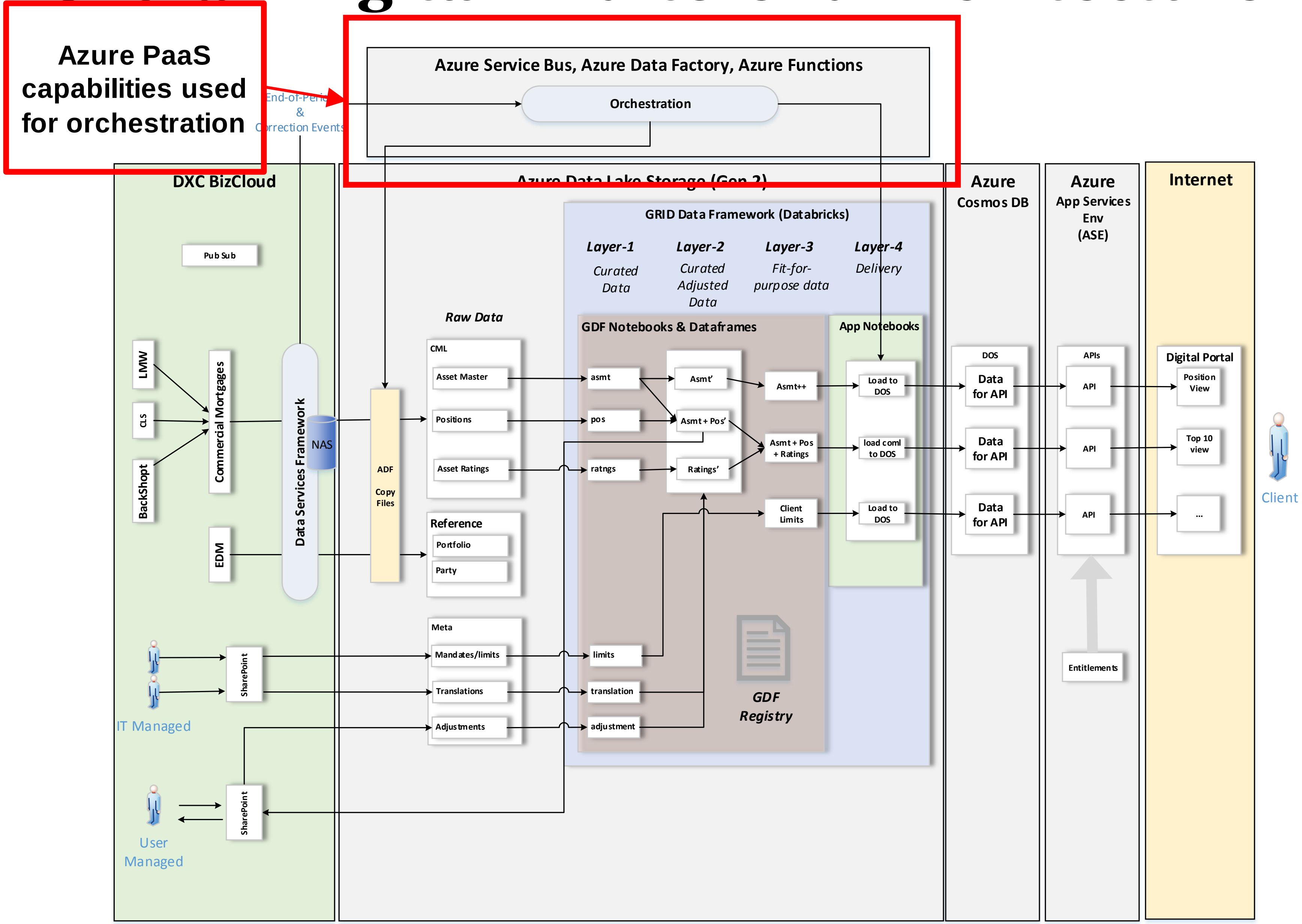


Overall Digital End-to-end Architecture

GRID use of the Azure Data Lake for long-term data storage



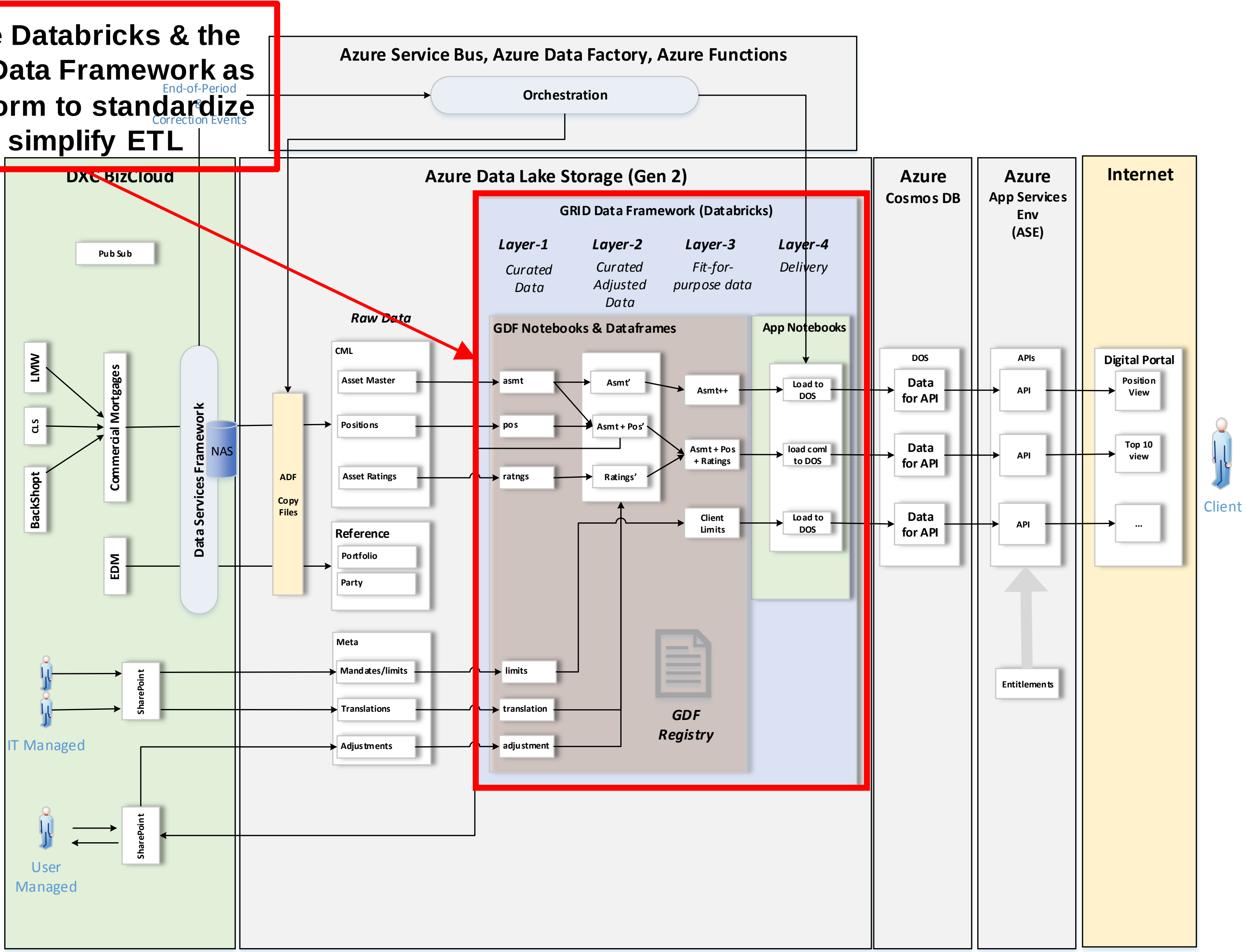
Overall Digital End-to-end Architecture



Azure PaaS capabilities used for orchestration

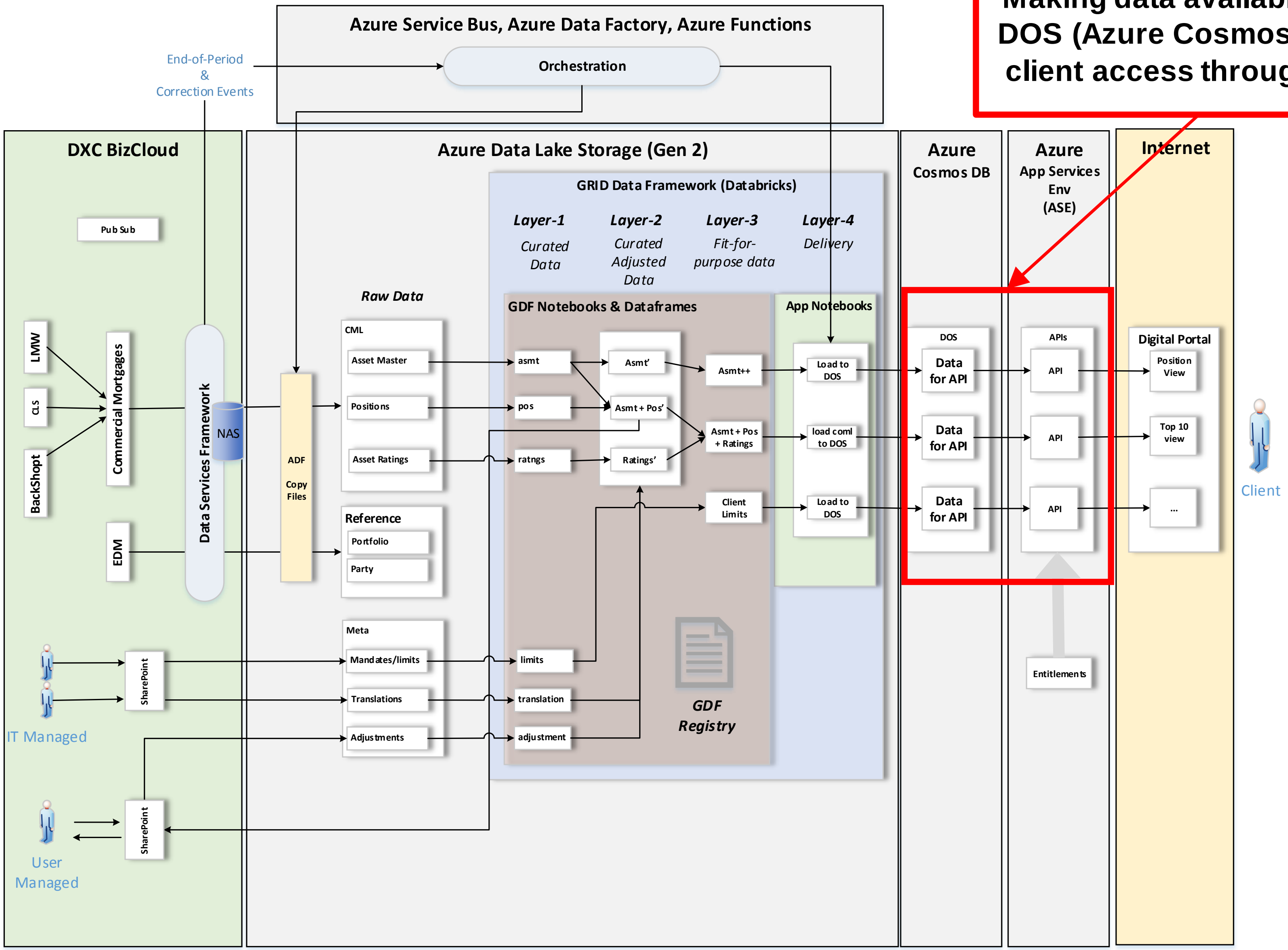
Overall Digital End-to-end Architecture

Azure Databricks & the GRID Data Framework as a Platform to standardize & simplify ETL

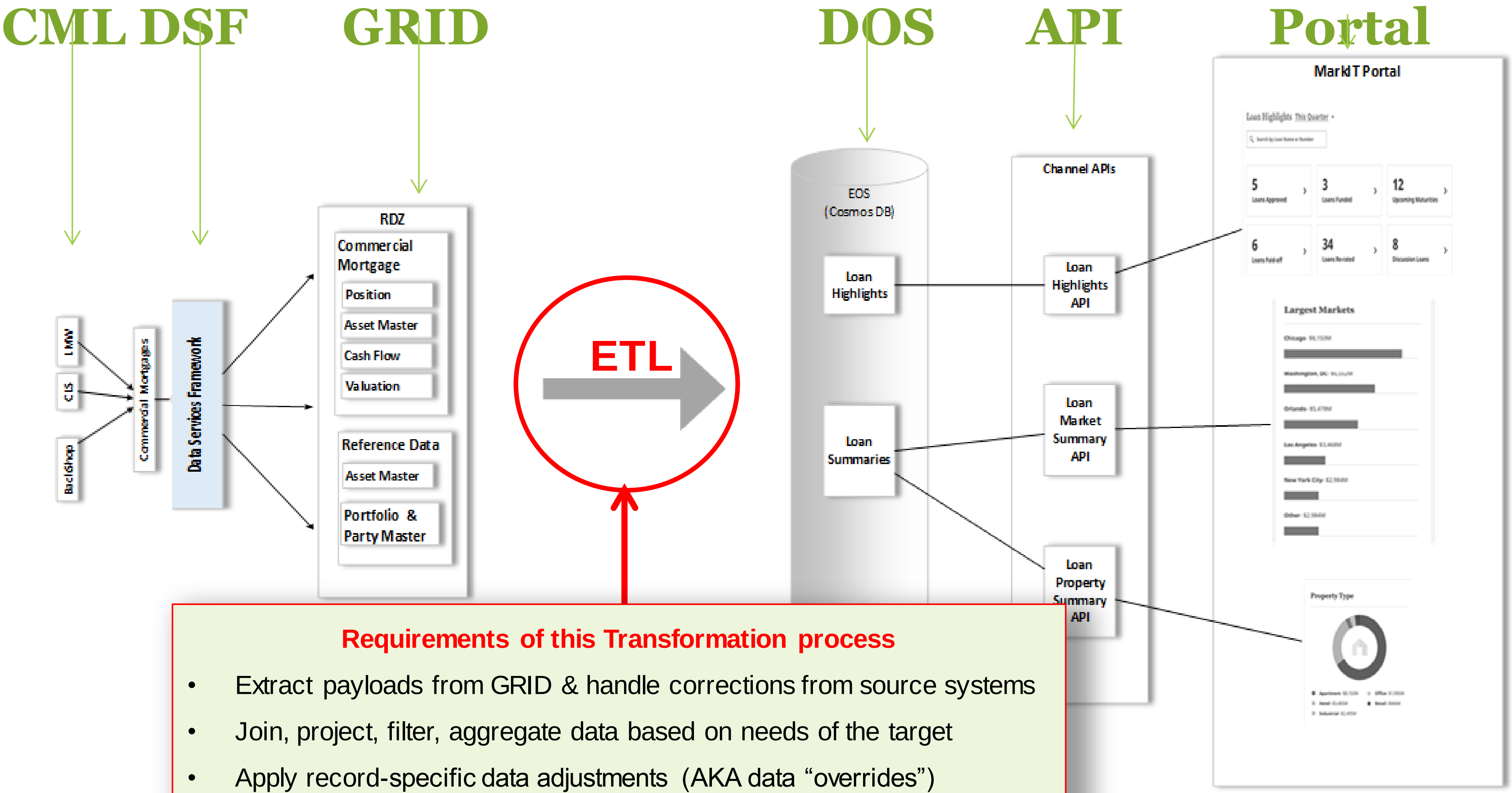


Overall Digital End-to-end Architecture

Making data available in the DOS (Azure Cosmos DB) for client access through APIs.



Problem: transformations from GRID to DOS



Requirements of this Transformation process

- Extract payloads from GRID & handle corrections from source systems
- Join, project, filter, aggregate data based on needs of the target
- Apply record-specific data adjustments (AKA data “overrides”)
- Apply “Langhorne-style” domain-value translations
- Load final results to DOS (for delivery to APIs and the Digital Portal)

MetLife

Solution

- Azure Databricks

- GRID Dataframe Framework (GDF)

ETL operations supported

Operation	Description
Select	Standard SQL projection to reduce the columns returned *e.g. select asset_id, asset_type from ...*
Filter	Standard SQL filter to reduce the rows returned *e.g. ... where asset_type == ‘CML’*
Join	Standard SQL joins across multiple data frames *e.g. select ... from AMster join Pos on Amster.id = Pos.id*
Aggregate	Standard SQL aggregations, sum, average, count, ... *e.g. sum the principal balance for all loans aggregated by property type*
Translate	Translate specific field values for all data, for a specific client *e.g. for client X, translate property type “Apt” to “Apartments” for all of their loans*
Override	Apply a specific adjustment to a specific record for a specific as-of-date *e.g. change the met-rating for loan with ID 12345 from “AA” to “A” , only on July-10-2019*
Reformat	Transform a value to a new format *e.g. format double value 7981234.3200001 to string “$7,981,234.32”*
Extract from source	Read canonical JSON data from the RDZ to create a dataframe
Load to target	Write a dataframe to some target data source, e.g. Cosmos DB

GRID - Global Repository for Investments Data

GRID

Defines the Standard Layout for data in Azure Data Lake Storage (ADLS Gen2)

GRID stores payload data delivered from source system applications

GRID Data organized by:

- Data Class, Data Set Type, As of Date, Region & Period

GRID Data is immutable:

- Data are never deleted
- Changes are delivered as **full payloads** or **correction payloads**
- All payloads are timestamped (in UTC time)

Any data in the GRID is found by knowing:

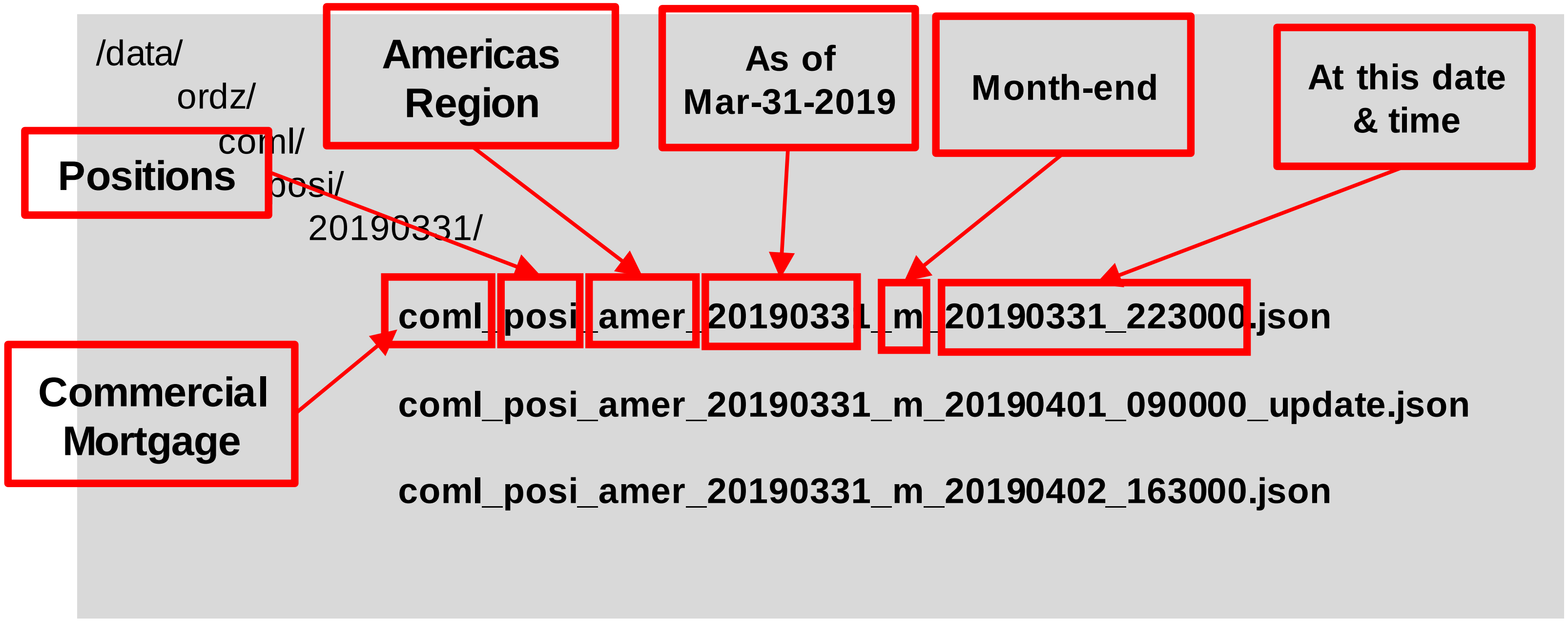
DataClass, DataSetType, As-of-Date, Region, Period, Point-in-time

GRID Example

Defines the Standard Layout for data in Azure Data Lake Storage (ADLS Gen2)

End-of- month Commercial Mortgage Positions for March 2019 can found in the folder: **`/data/ordz/coml/posi/20190331/`**

The folder contains all versions of that data, including all changes



/data/
ordz/
coml/
posi/
20190331/

coml_posi_amer_20190331_m_20190331_223000.json

coml_posi_amer_20190331_m_20190401_090000_update.json

coml_posi_amer_20190331_m_20190402_163000.json

GRID Example

Defines the Standard Layout for data in Azure Data Lake Storage (ADLS Gen2)

The folder contains all versions of that data, including all changes

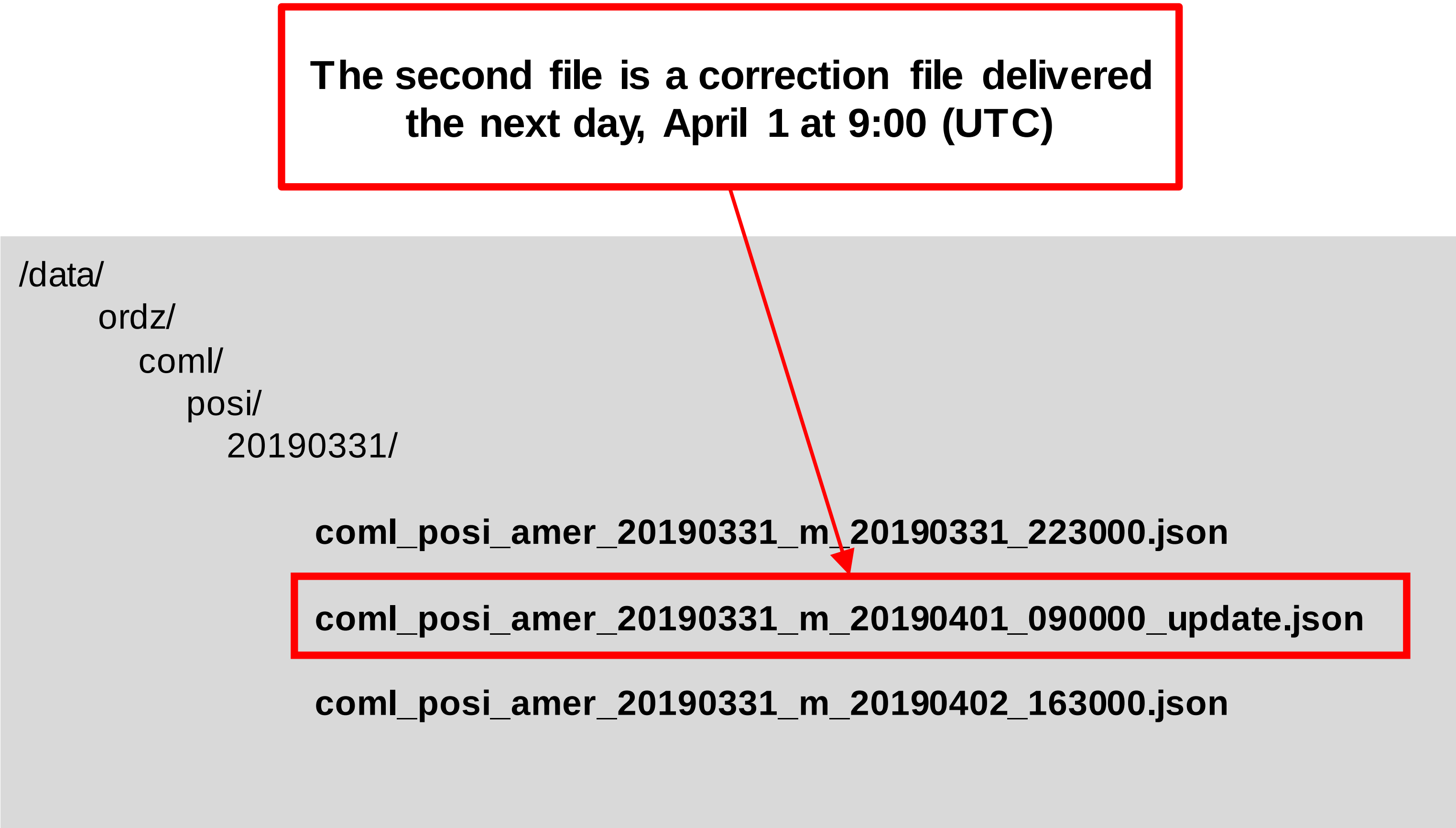
The first file is the end-of-month payload delivered on March 31 at 22:30 (UTC)

```
/data/
      ordz/
         coml/
            posi/
               20190331/

                      coml_posi_amer_20190331_m_20190331_223000.json

                      coml_posi_amer_20190331_m_20190401_090000_update.json

                      coml_posi_amer_20190331_m_20190402_163000.json
```

GRID Example

Defines the Standard Layout for data in Azure Data Lake Storage (ADLS Gen2)

The folder contains all versions of that data, including all changes



The second file is a correction file delivered the next day, April 1 at 9:00 (UTC)

```
/data/
        ordz/
            coml/
                posi/
                    20190331/

                            coml_posi_amer_20190331_m_20190331_223000.json

                            coml_posi_amer_20190331_m_20190401_090000_update.json

                            coml_posi_amer_20190331_m_20190402_163000.json
```

GRID Example

Defines the Standard Layout for data in Azure Data Lake Storage (ADLS Gen2)

The folder contains all versions of that data, including all changes

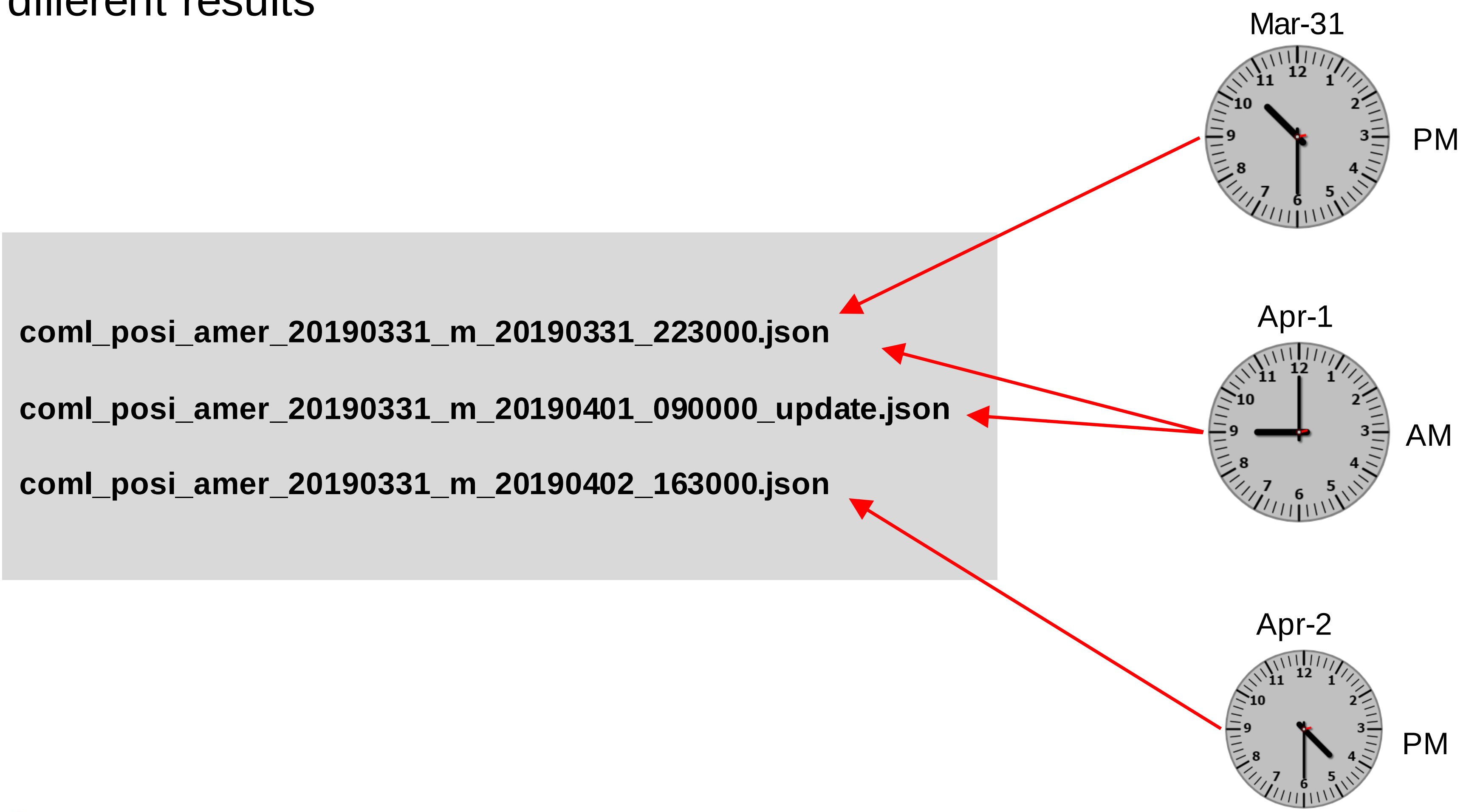
The third file is full replacement of the March month-end file delivered on April 2 at 16:30 (UTC)

```
/data/
        ordz/
            coml/
                posi/
                    20190331/

                            coml_posi_amer_20190331_m_20190331_223000.json

                            coml_posi_amer_20190331_m_20190401_090000_update.json

                            coml_posi_amer_20190331_m_20190402_163000.json
```

Logic for consuming ORDZ Files

data depends on the “point in time” that you consume it

Depending on when you consume the ORDZ Data, you may get different results



Mar-31

PM

coml_posi_amer_20190331_m_20190331_223000.json

Apr-1

AM

coml_posi_amer_20190331_m_20190401_090000_update.json

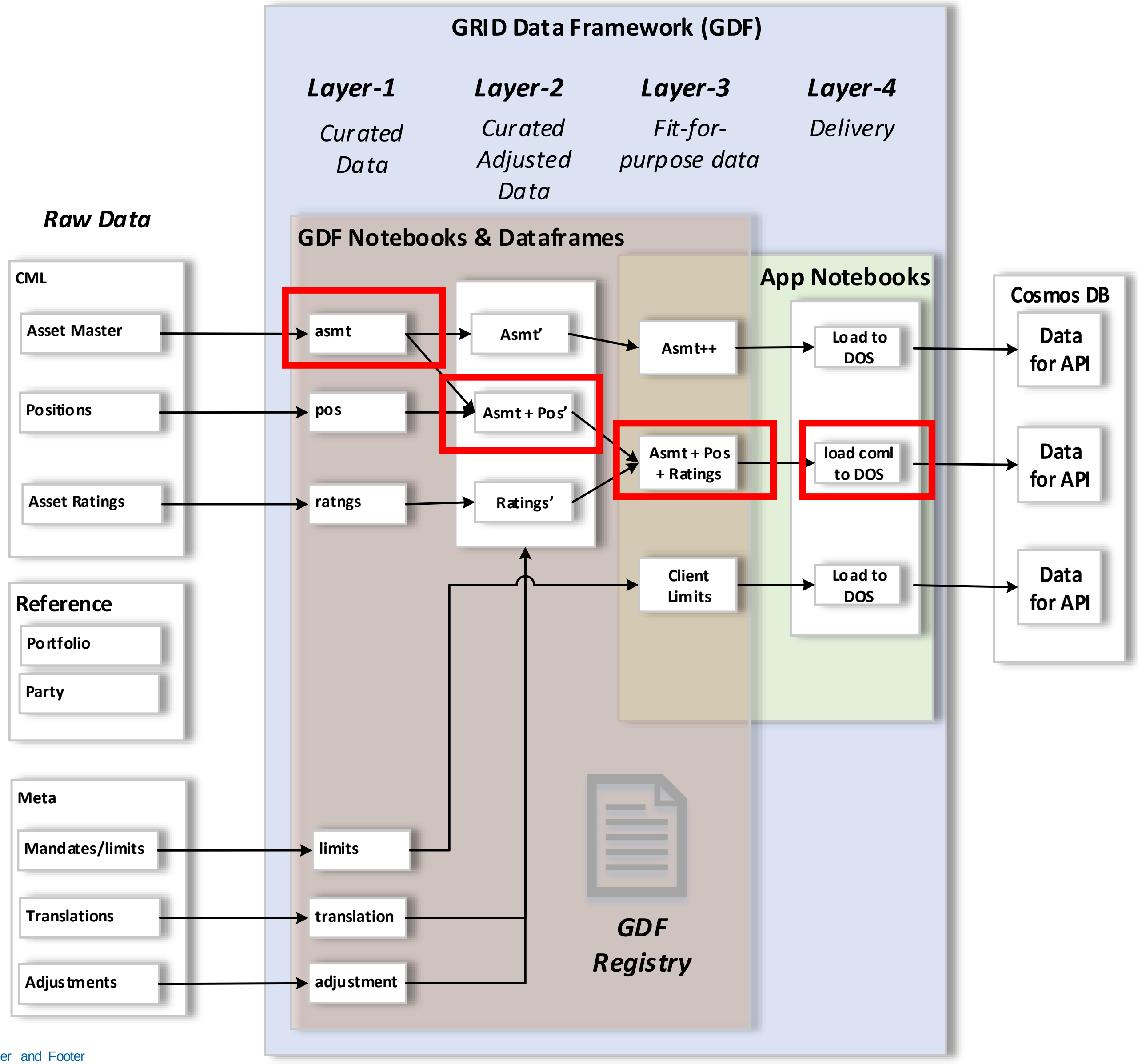
coml_posi_amer_20190331_m_20190402_163000.json

Apr-2

PM

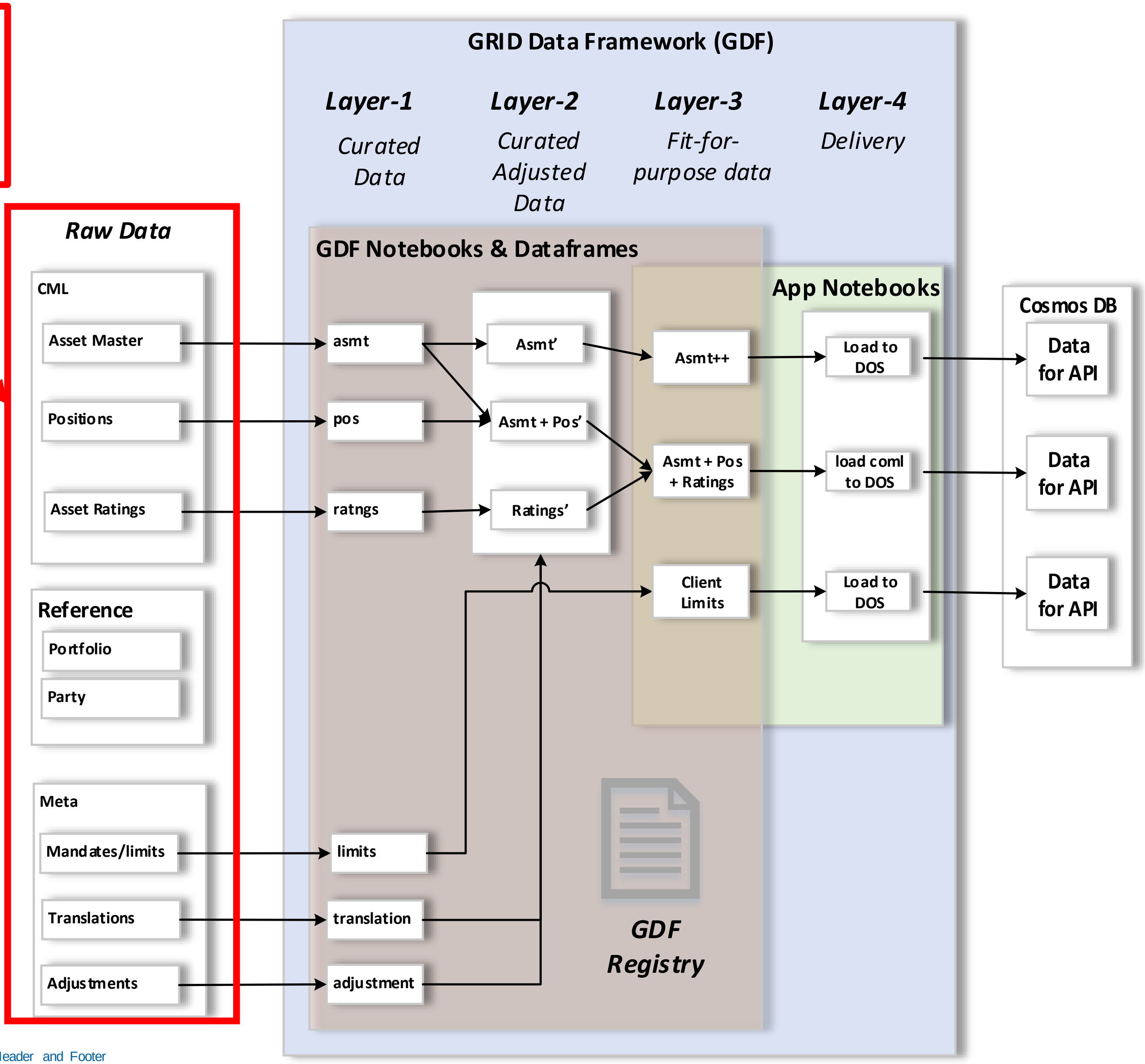
GRID Data Framework (GDF)

GRID Data Framework – a layered approach to curating data from source to target



GRID Data Framework – source system data files

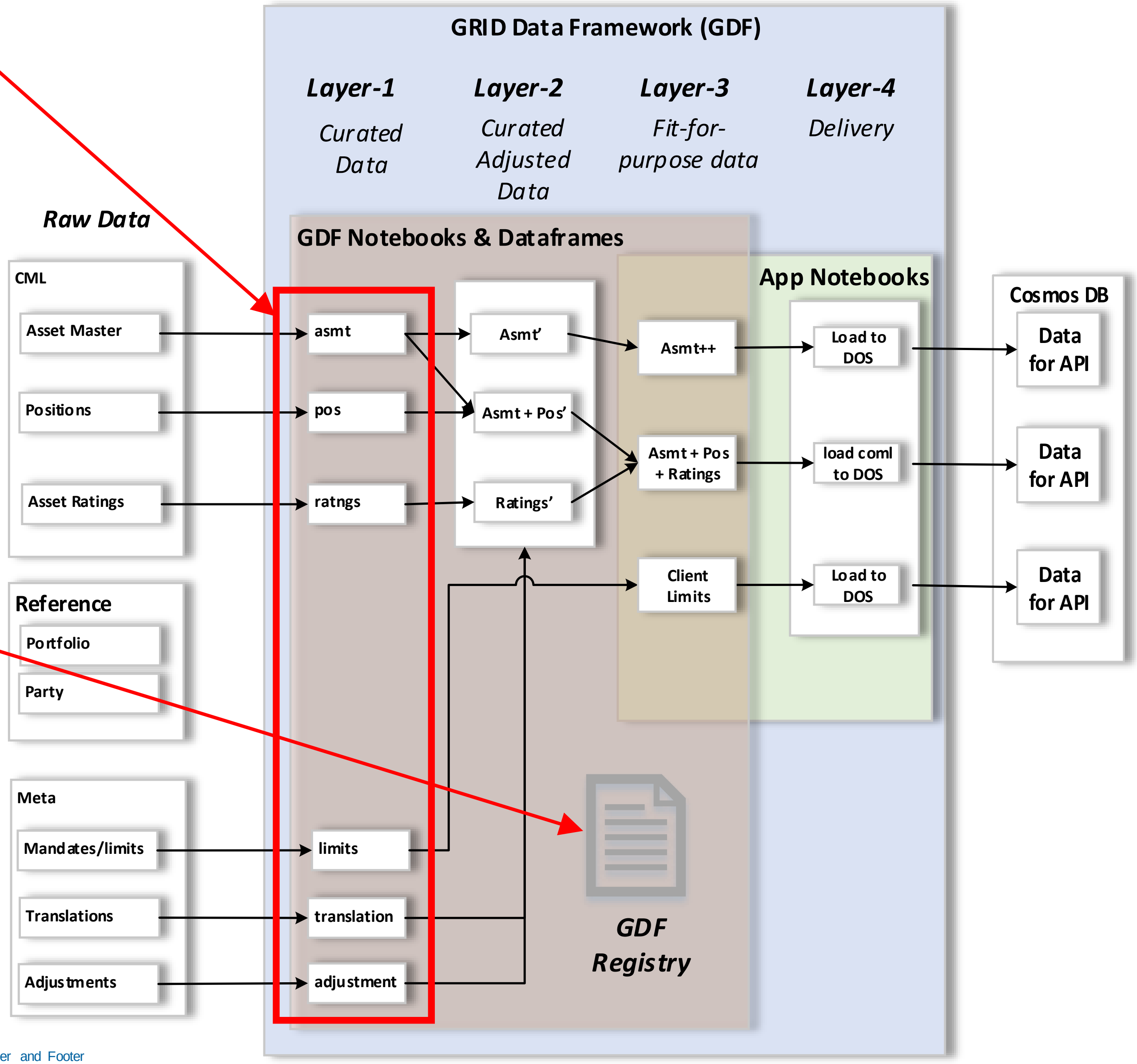
These are the raw data files produced by the source systes



GRID DataFrames – created automatically from the GRID

These are the GRID Dataframes

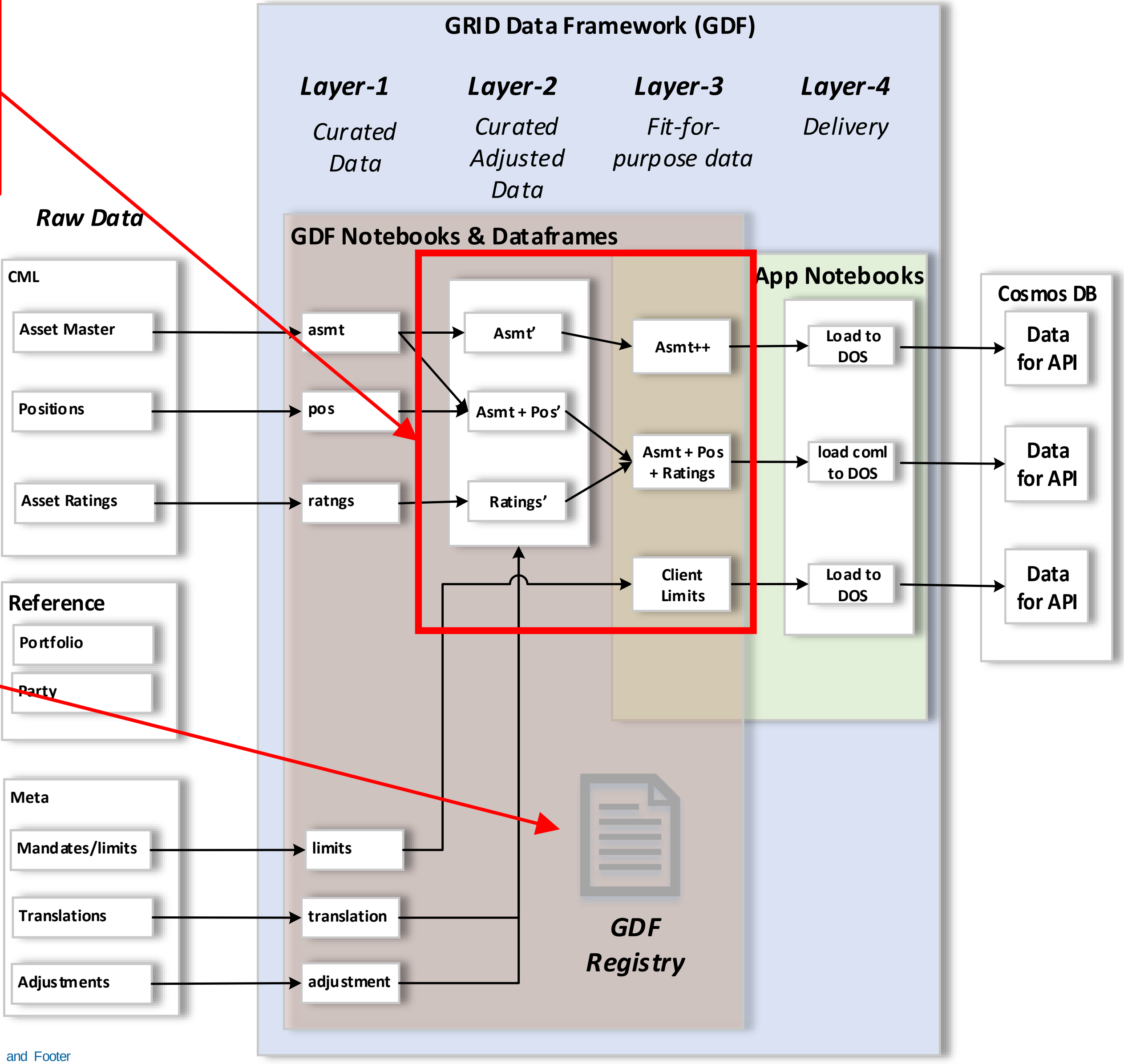
GRID Dataframes are automatically created, based on meta-data in the GDF Registry



Derived DataFrames

These are the Derived Dataframes

The GDF Registry describes how one dataframe is derived from others

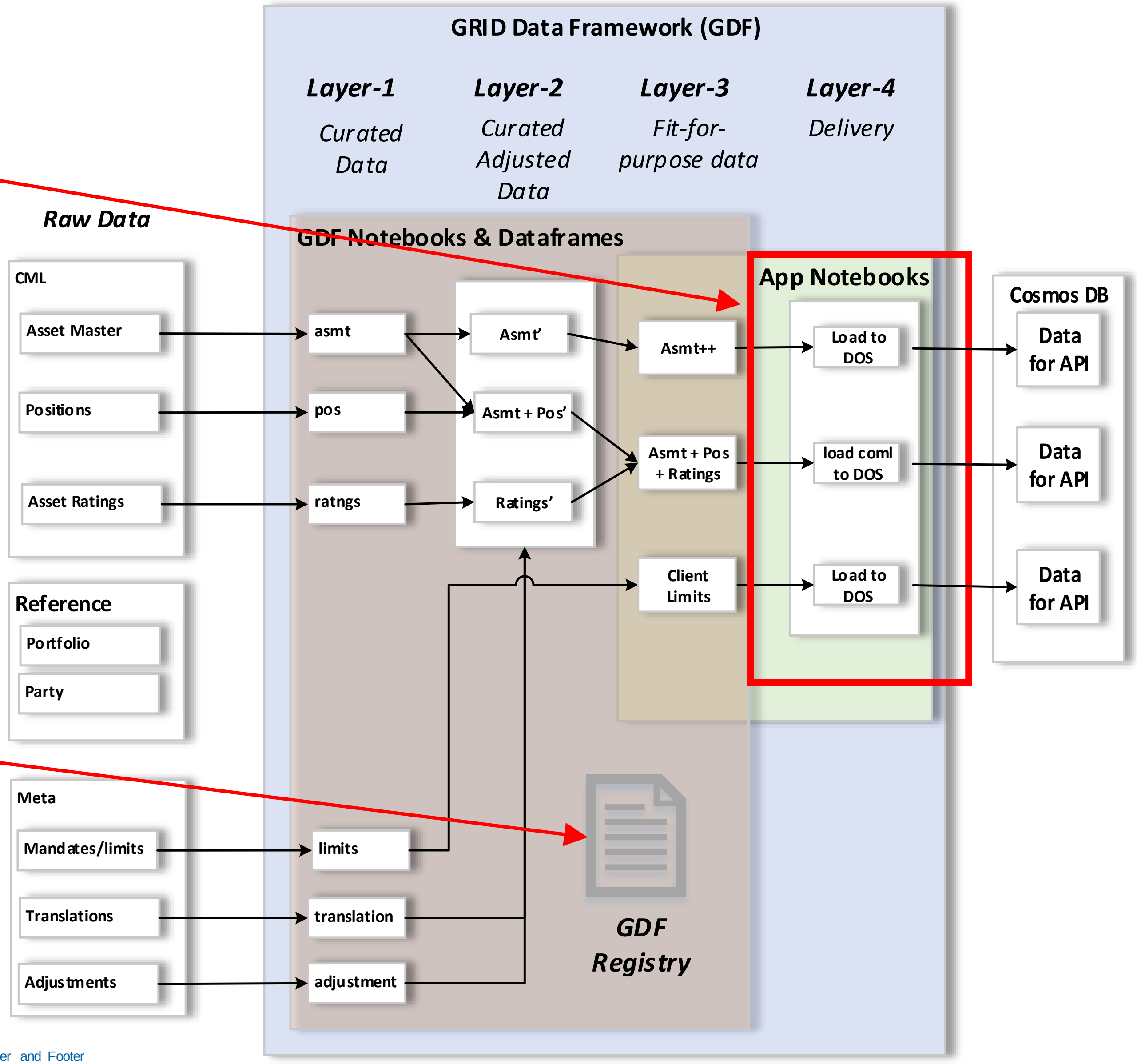


App Notebooks

These are the App Notebooks

App Notebooks are used to load data to a target.

App Notebooks are also registered in the GDF Registry.



Features of the GRID Data Framework

- GRID dataframes **created automatically** from the GRID
- Registry makes **all data dependencies and lineage explicit**
- Design enables **time-travel view of data** from any point in time
- **curated data is automatically persisted** in Data Lake
- **Data translations applied automatically** from meta-data
- **Data adjustments applied automatically** from meta-data
- Defines a **standard pattern** for developing GRID applications

GRID Data Framework Registry

GRID Data Framework Registry

GDF Registry maintains information about:

- *GRID dataframes* - are dataframes that are created automatically from the data in the GRID, whenever one attempts to retrieve the dataframe.

- *Derived dataframes* - are dataframes that are created by a notebook by consuming data from other dataframes, and then using that data to produce a new dataframe

- *App notebooks* - are those notebooks that teams would develop in order to access the data in either GRID dataframes and/or derived dataframes in order to produce a result that can be delivered to another target and for another purpose . App notebooks will not be allowed to create dataframes that other apps are able to consume. If an app has a desire to create a dataframe that might be sharable with other apps, then that capability should be promoted to a derived dataframe.

GRID Dataframe Registry

Simplify consumption of data from the GRID

GRID DataFrame Registry

automatically created using registry meta-data

Example GRID DataFrame Registry definitions:

Name of the DataFrame

Where to find the data in the GRID

- Definition of Commercial Loan Asset Master DataFrame

```
{
  "name": "coml_asmt_df",
  “grid": {“dataClass":"coml", " dataSetType ":"asmt"},
  "description": “Commercial Loan asset master data"
}
```

- Definition of Commercial Loan Positions Dataframe

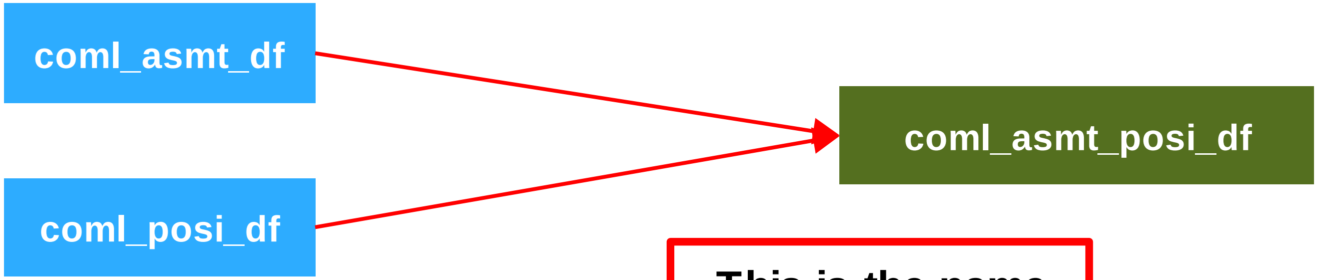
```
{
  "name": "coml_posi_df",
  “grid": {“dataClass":"coml", “dataSetType":“posi"},
  "description": “Commercial Loan Positions data"
}
```

Derived Dataframe Registry

The result of combining dataframes together

Derived DataFrames

registry meta-data defines how they are created



This is the name of the DataFrame

This specifies how this derived dataframe depends on data from other dataframes

Example Derived DataFrame Registry definitions:

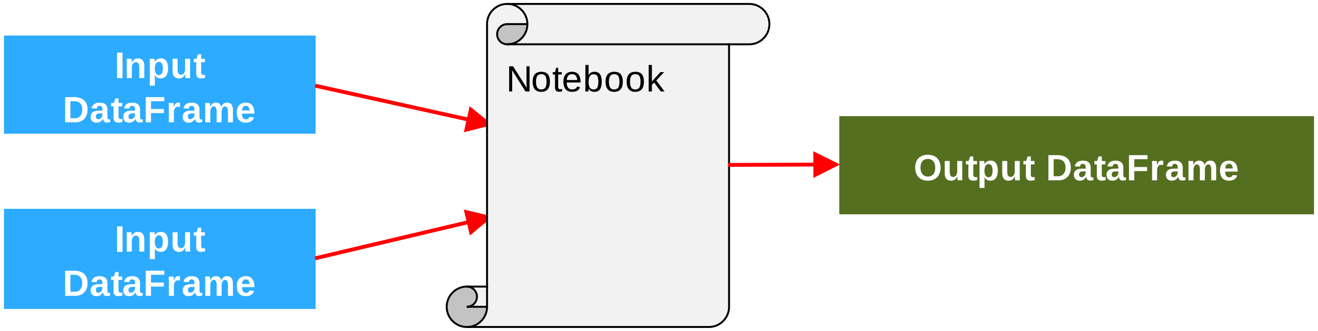
- Definition of Asset Position DataFrame

```
{
  "name": "coml_asmt_posi_df",
  "dependsOn": ["coml_asmt_df", "coml_posi_df"],
  “Notebook":"/Shared/dff/ordz/coml_asmt_posi_df",
  “description” : “commercial loan positions with asset data joined”
},
```

GDF Notebooks

A GDF Notebook defines a function

Transforms input dataframe(s) to an output dataframe



Notebooks

Where we write the logic

Sample Notebook to create a dataframe

Is where you specify how to create the derived dataframe

Example Derived DataFrame Registry definition

```
from dff.dfnotebook import GDFContext
...
nb = GDFContext(‘coml_asmt_posi_df’)

asmt = nb.get_df(‘coml_asmt_df’)
posi = nb.get_df(‘coml_posi_df’)

joined = sql(‘select * from ‘ + asmt.view + ‘ a, ‘
                              + posi.view + ‘p, ’
              ‘where a.MetLifeAssetID = p.MetLifeAssetID’)



nb.store_curated_df(joined)
```

GDF Notebook Definition

Is where you specify how to create the derived dataframe

Example Derived DataFrame Registry definition

```
from dff.dfnotebook import GDFContext

nb = GDFContext(‘coml_asmt_posi_df’)

asmt = nb.get_df(‘coml_asmt_df’)
posi = nb.get_df(‘coml_posi_df’)

joined = sql(‘select * from ‘ + asmt.view + ‘ a, ‘
                               + posi.view + ‘p, ’
              ‘where a.MetLifeAssetID = p.MetLifeAssetID’)


nb.store_curated_df(joined)
```

This creates the GDF Context Object

Here we retrieve the asset master and Positions GRID Dataframes

Use Spark-SQL to join the 2 dataframes

Finally persist the result to be returned to whomever requested the data

App Notebooks

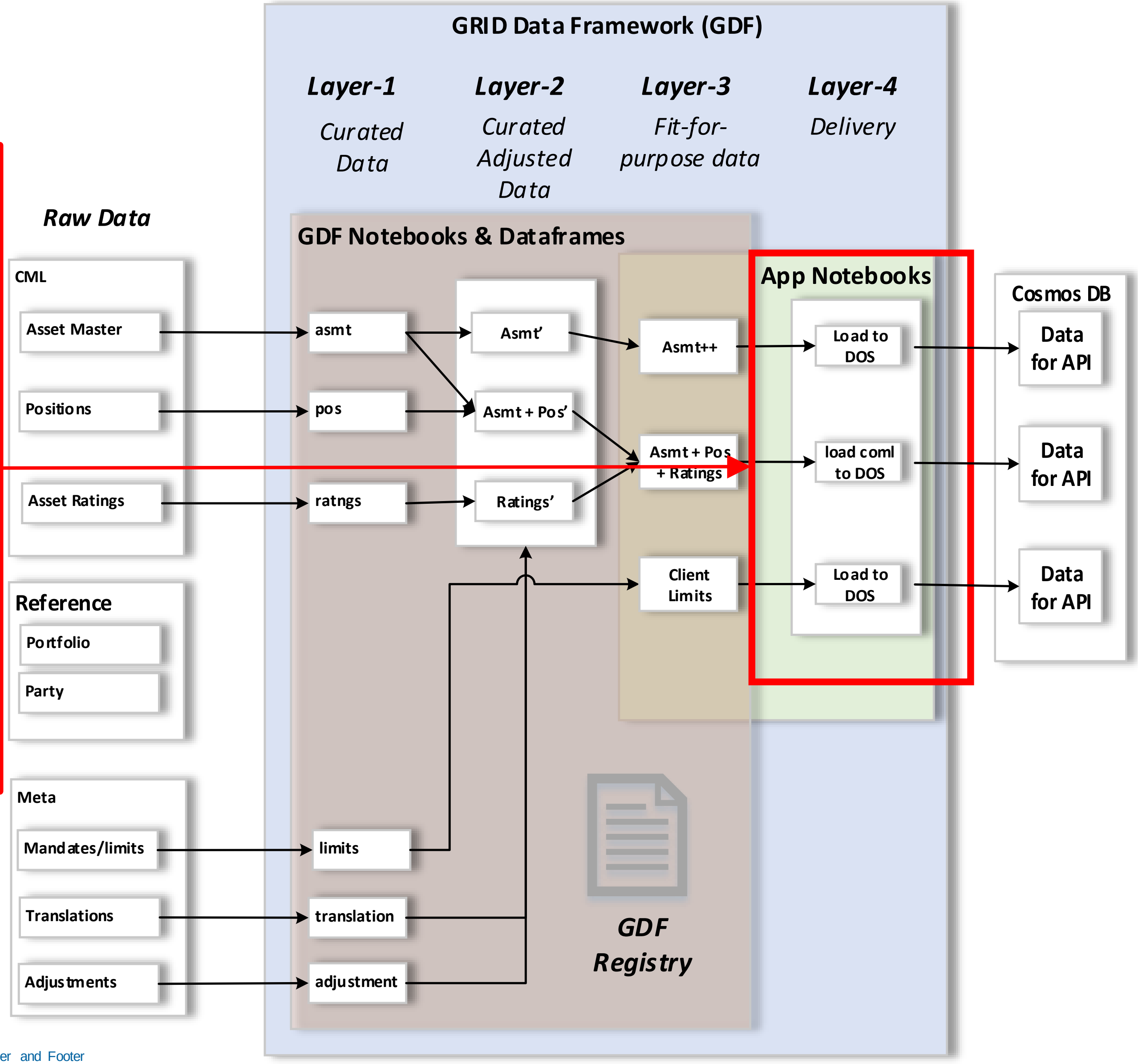
Use App Notebooks to consume data for an application specific purpose

App Notebooks

These are the App Notebooks

App Notebooks are the last layer of Notebooks

An App Notebook's job is to consume GRID Dataframes and then deliver the result to a target.



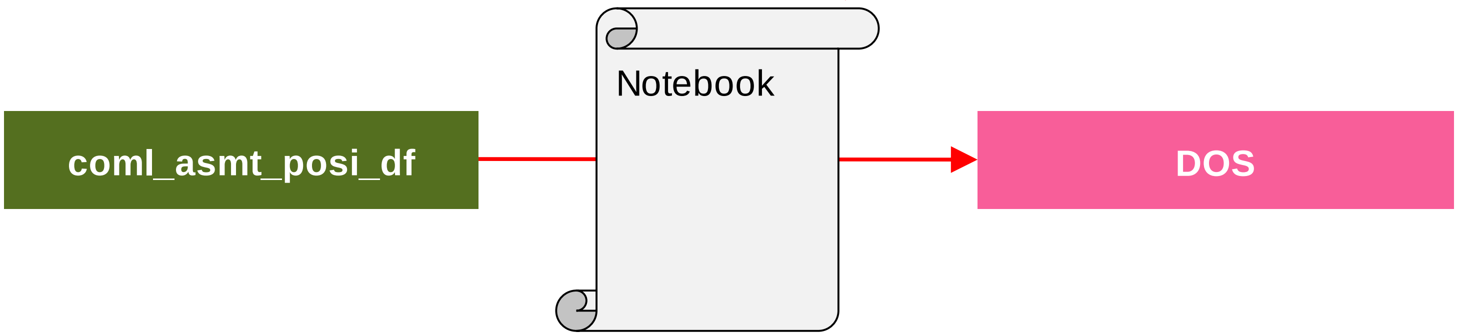
App Notebook Registry

Example App Notebook registry:

```
{
   "name": “load_loan_list_to_dos",
   "dependsOn": ["coml_asmt_posi_df"],
  “notebook":"/Shared/dff/ordz/load_coml_asmt_posi_to_dos,
  “description” : “app to load asmt_posi to dos”
},
```

This specifies the “code” to run to consume data in order to load to some target

The code is in Databricks App Notebook



GDF App Notebook Definition

App Notebook to retrieve data from a DataFrame and then write to a target

Example of writing loan data to Cosmos DB

```
# retrieve the asset master positions data
asmt_posi = nb.get_df(‘coml_asmt_posi_df’)

# Write the new asmt_posi dataframe to Cosmos DB
asmt_posi.write  \
     .format("com.microsoft.azure.cosmosdb.spark")  \
     .options(**writeConfig)  \
     .save()
```

***writeConfig* specifies the instance of Cosmos DB to connect & provides credentials**

In conclusion…

Quick animation of the end-to-end flow

Link to PDF

Digital Architecture -> Paradigm shift

Current	Future
XML	JSON
MSSQL Database Table	Spark DataFrame
SQL	Spark SQL
Stored Procedure	Databricks Notebook
T-SQL	Python, PySpark, Spark SQL
NAS	Azure Data Lake Storage (Gen 2)
TWS Jobs Scheduler	Events + Pub/Sub
Load by parsing XML	Read a GRID DataFrame

Appendix

Using the GDF to apply Data Adjustments and Data Translations

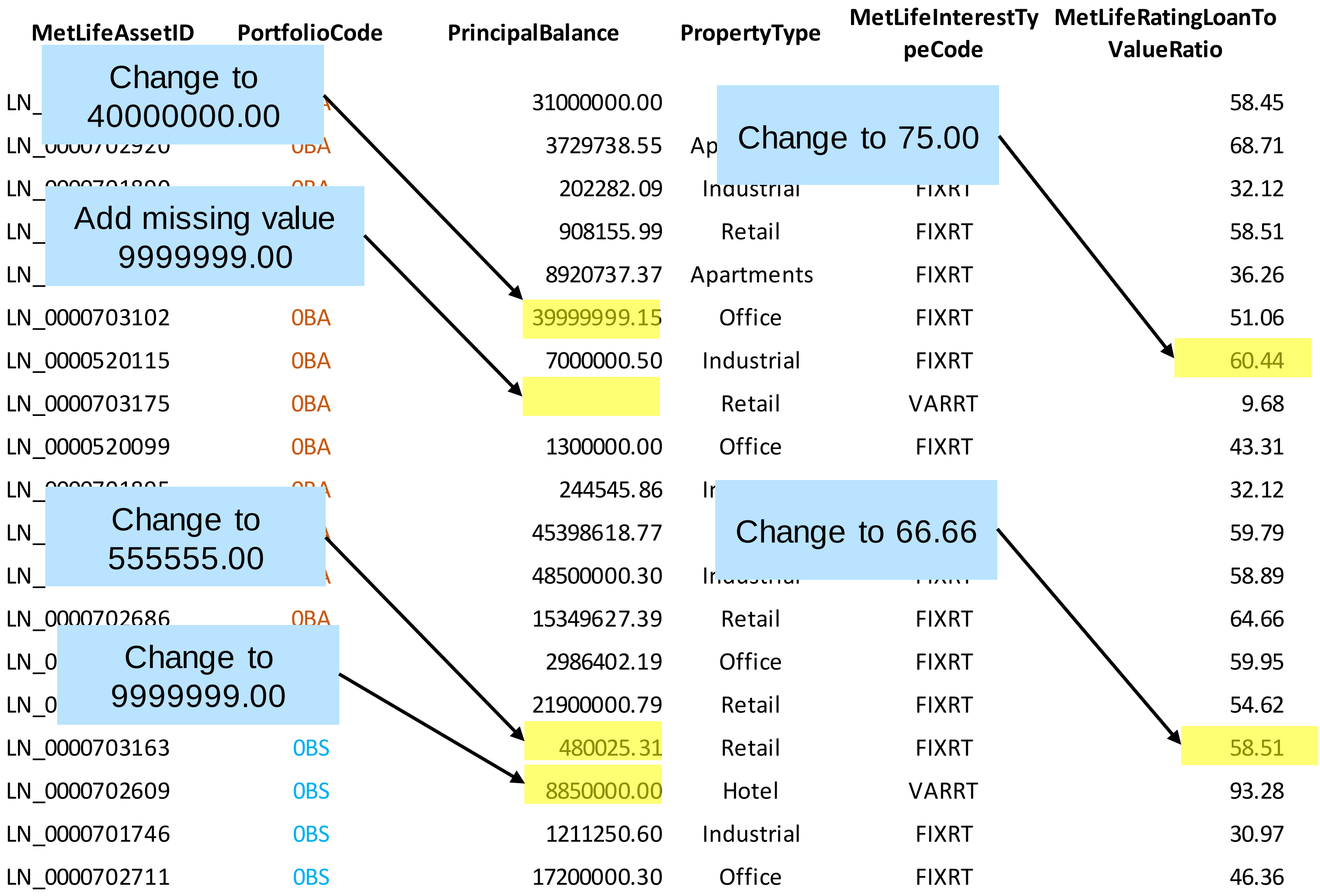
Sample Report to be adjusted/translated

Subset of commercial loan asset position records, extracted to a CSV file

MetLifeAssetID	PortfolioCode	PrincipalBalance	PropertyType	MetLifeInterestTypeCode	MetLifeRatingLoanToValueRatio
LN_0000703144	0BA	31000000.00	Office	FIXRT	58.45
LN_0000702920	0BA	3729738.55	Apartments	FIXRT	68.71
LN_0000701890	0BA	202282.09	Industrial	FIXRT	32.12
LN_0000703163	0BA	908155.99	Retail	FIXRT	58.51
LN_0000702359	0BA	8920737.37	Apartments	FIXRT	36.26
LN_0000703102	0BA	39999999.15	Office	FIXRT	51.06
LN_0000520115	0BA	7000000.50	Industrial	FIXRT	60.44
LN_0000703175	0BA		Retail	VARRT	9.68
LN_0000520099	0BA	1300000.00	Office	FIXRT	43.31
LN_0000701895	0BA	244545.86	Industrial	FIXRT	32.12
LN_0000702521	0BA	45398618.77	Office	FIXRT	59.79
LN_0000520120	0BA	48500000.30	Industrial	FIXRT	58.89
LN_0000702686	0BA	15349627.39	Retail	FIXRT	64.66
LN_0000702708	0BA	2986402.19	Office	FIXRT	59.95
LN_0000702755	0BS	21900000.79	Retail	FIXRT	54.62
LN_0000703163	0BS	480025.31	Retail	FIXRT	58.51
LN_0000702609	0BS	8850000.00	Hotel	VARRT	93.28
LN_0000701746	0BS	1211250.60	Industrial	FIXRT	30.97
LN_0000702711	0BS	17200000.30	Office	FIXRT	46.36

Example data adjustments

Adjustments are changes made to specific records – this requires a "key" to identify the specific records – in this example, MetLifeAssetID & PortfolioCode are the keys



MetLifeAssetID	PortfolioCode	PrincipalBalance	PropertyType	MetLifeInterestTypeCode	MetLifeRatingLoanToValueRatio
LN_	A	31000000.00			58.45
LN_0000702920	0BA	3729738.55	Ap		68.71
LN_	0BA	202282.09	Industrial	FIXRT	32.12
LN_		908155.99	Retail	FIXRT	58.51
LN_		8920737.37	Apartments	FIXRT	36.26
LN_0000703102	0BA	39999999.15	Office	FIXRT	51.06
LN_0000520115	0BA	7000000.50	Industrial	FIXRT	60.44
LN_0000703175	0BA		Retail	VARRT	9.68
LN_0000520099	0BA	1300000.00	Office	FIXRT	43.31
LN_	0BA	244545.86	In		32.12
LN_	A	45398618.77			59.79
LN_	A	48500000.30	Industrial	FIXRT	58.89
LN_0000702686	0BA	15349627.39	Retail	FIXRT	64.66
LN_0		2986402.19	Office	FIXRT	59.95
LN_0		21900000.79	Retail	FIXRT	54.62
LN_0000703163	0BS	480025.31	Retail	FIXRT	58.51
LN_0000702609	0BS	8850000.00	Hotel	VARRT	93.28
LN_0000701746	0BS	1211250.60	Industrial	FIXRT	30.97
LN_0000702711	0BS	17200000.30	Office	FIXRT	46.36

Change to 40000000.00

Add missing value 9999999.00

Change to 555555.00

Change to 9999999.00

Change to 75.00

Change to 66.66

CSV file to specify data adjustments

Use these columns to specify the keys to identify the specific records

These columns are used to specify the “adjustment”

These columns define the “as of date” when the adjustment is applicable

Key:MetLifeAssetID	Key:PortfolioCode	PrincipalBalance	MetLifeRatingLoanToValueRatio	StartDate	EndDate
LN_0000703102	0BA	40000000		8/14/2019	9/1/2019
LN_0000520115	0BA		75	8/14/2019	9/1/2019
LN_0000703175	0BA	9999999		8/14/2019	9/1/2019
LN_0000703163	0BS	555555	66.66	8/14/2019	9/1/2019
LN_0000702609	0BS	9999999		8/14/2019	9/1/2019

Sample Report after applying data adjustments

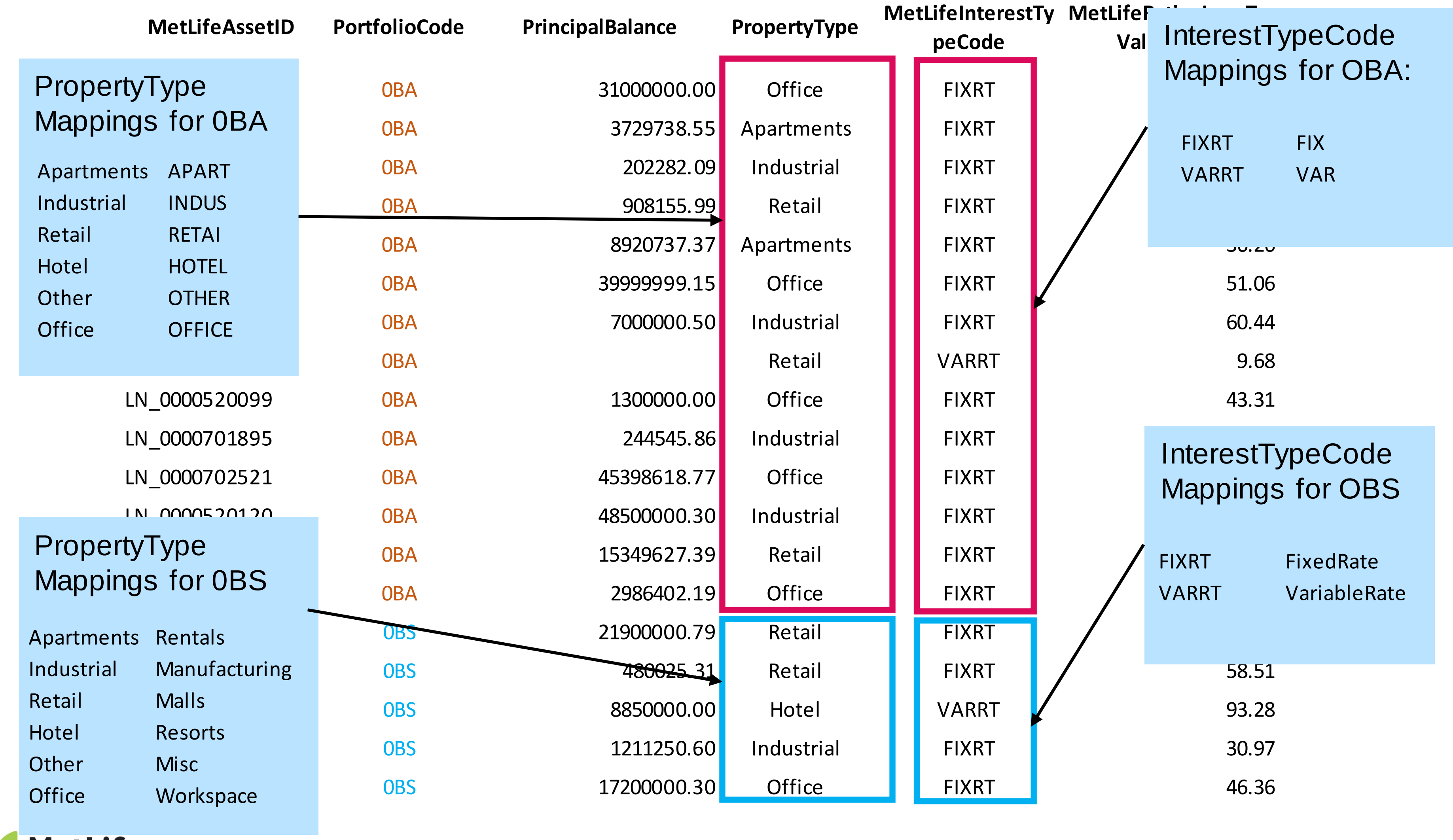
Adjusted values are depicted in **red**

MetLifeAssetID	PortfolioCode	PrincipalBalance	PropertyType	MetLifeInterestTypeCode	MetLifeRatingLoanToValueRatio
LN_0000703144	0BA	31000000.00	Office	FIXRT	58.45
LN_0000702920	0BA	3729738.55	Apartments	FIXRT	68.71
LN_0000701890	0BA	202282.09	Industrial	FIXRT	32.12
LN_0000703163	0BA	908155.99	Retail	FIXRT	58.51
LN_0000702359	0BA	8920737.37	Apartments	FIXRT	36.26
LN_0000703102	0BA	**40000000.00**	Office	FIXRT	51.06
LN_0000520115	0BA	7000000.50	Industrial	FIXRT	**75.00**
LN_0000703175	0BA	**9999999.00**	Retail	VARRT	9.68
LN_0000520099	0BA	1300000.00	Office	FIXRT	43.31
LN_0000701895	0BA	244545.86	Industrial	FIXRT	32.12
LN_0000702521	0BA	45398618.77	Office	FIXRT	59.79
LN_0000520120	0BA	48500000.30	Industrial	FIXRT	58.89
LN_0000702686	0BA	15349627.39	Retail	FIXRT	64.66
LN_0000702708	0BA	2986402.19	Office	FIXRT	59.95
LN_0000702755	0BS	21900000.79	Retail	FIXRT	54.62
LN_0000703163	0BS	**555555.00**	Retail	FIXRT	**66.66**
LN_0000702609	0BS	**9999999.00**	Hotel	VARRT	93.28
LN_0000701746	0BS	1211250.60	Industrial	FIXRT	30.97
LN_0000702711	0BS	17200000.30	Office	FIXRT	46.36

Data Translations

Example data translations

Translations are “domain value” mappings that apply to many records – translations change a field from having one value to some other value, based on a mapping



MetLifeAssetID	PortfolioCode	PrincipalBalance	PropertyType	MetLifeInterestTypeCode	MetLifeR Val
	0BA	31000000.00	Office	FIXRT	
	0BA	3729738.55	Apartments	FIXRT	
	0BA	202282.09	Industrial	FIXRT	
	0BA	908155.99	Retail	FIXRT	
	0BA	8920737.37	Apartments	FIXRT	
	0BA	39999999.15	Office	FIXRT	51.06
	0BA	7000000.50	Industrial	FIXRT	60.44
	0BA		Retail	VARRT	9.68
LN_0000520099	0BA	1300000.00	Office	FIXRT	43.31
LN_0000701895	0BA	244545.86	Industrial	FIXRT	
LN_0000702521	0BA	45398618.77	Office	FIXRT	
LN_0000520120	0BA	48500000.30	Industrial	FIXRT	
	0BA	15349627.39	Retail	FIXRT	
	0BA	2986402.19	Office	FIXRT	
	0BS	21900000.79	Retail	FIXRT	
	0BS	480025.31	Retail	FIXRT	58.51
	0BS	8850000.00	Hotel	VARRT	93.28
	0BS	1211250.60	Industrial	FIXRT	30.97
	0BS	17200000.30	Office	FIXRT	46.36

PropertyType Mappings for 0BA

Apartments	APART
Industrial	INDUS
Retail	RETAI
Hotel	HOTEL
Other	OTHER
Office	OFFICE

PropertyType Mappings for 0BS

Apartments	Rentals
Industrial	Manufacturing
Retail	Malls
Hotel	Resorts
Other	Misc
Office	Workspace

InterestTypeCode Mappings for OBA:

FIXRT	FIX
VARRT	VAR

InterestTypeCode Mappings for OBS

FIXRT	FixedRate
VARRT	VariableRate

CSV file to specify Data Translations

Key:PortfolioCode	Attribute	From	To	StartDate	EndDate
0BA	PropertyType	Apartments	APART	8/14/2019	
0BA	PropertyType	Industrial	INDUS	8/14/2019	
0BA	PropertyType	Retail	RETAI	8/14/2019	
0BA	PropertyType	Hotel	HOTEL	8/14/2019	
0BA	PropertyType	Other	OTHER	8/14/2019	
0BA	PropertyType	Office	OFFICE	8/14/2019	
0BA	MetLifeInterestTypeCode	FIXRT	FIX	8/14/2019	
0BA	MetLifeInterestTypeCode	VARRT	VAR	8/14/2019	
0BS	PropertyType	Apartments	Rentals	8/14/2019	8/31/2019
0BS	PropertyType	Industrial	Manufacturing	8/14/2019	8/31/2019
0BS	PropertyType	Retail	Malls	8/14/2019	8/31/2019
0BS	PropertyType	Hotel	Resorts	8/14/2019	8/31/2019
0BS	PropertyType	Other	Misc	8/14/2019	8/31/2019
0BS	PropertyType	Office	Workspace	8/14/2019	8/31/2019
0BS	MetLifeInterestTypeCode	FIXRT	FixedRate	8/14/2019	8/31/2019
0BS	MetLifeInterestTypeCode	VARRT	VariableRate	8/14/2019	8/31/2019

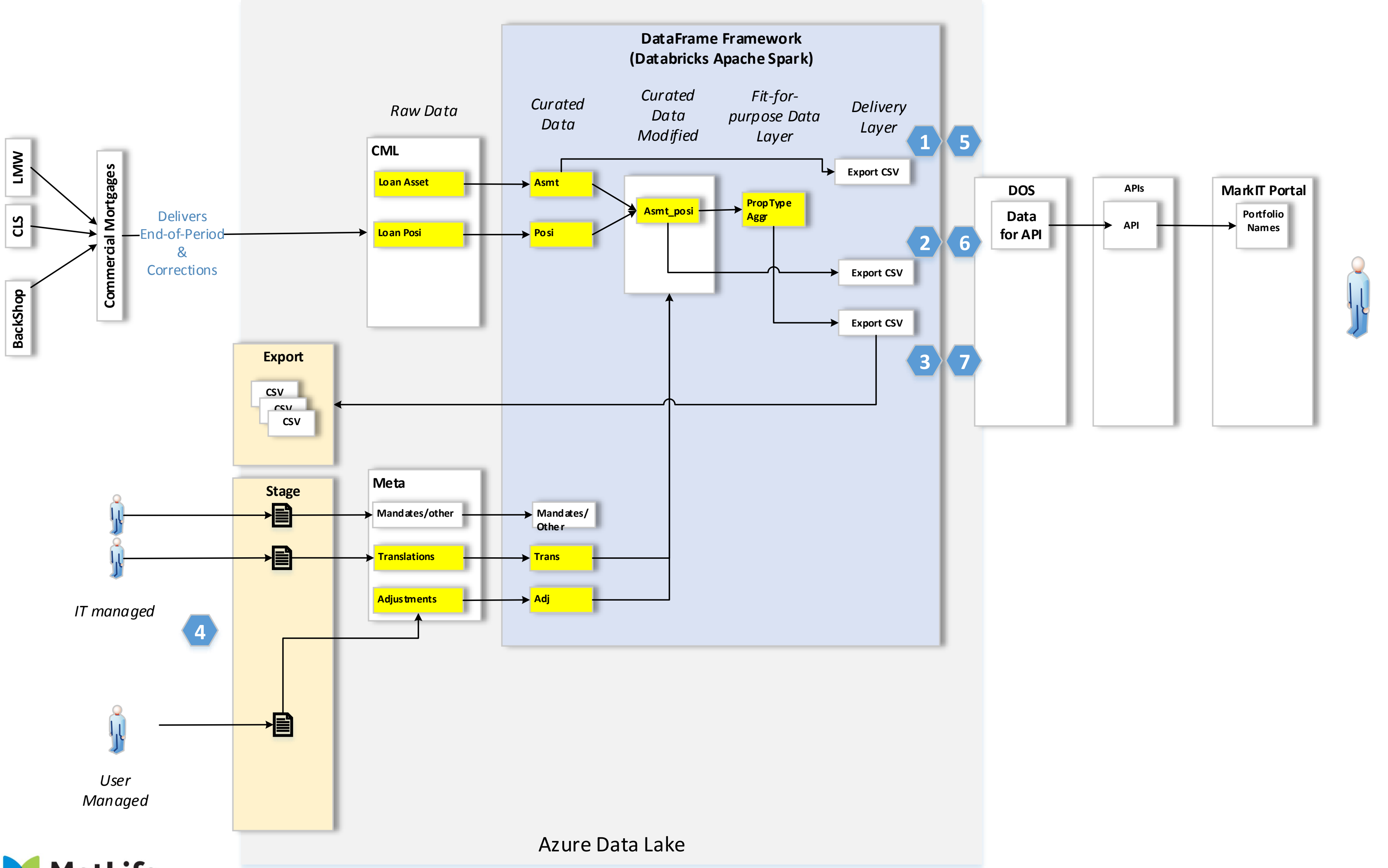
Sample Report after applying both Adjustments & Translations

MetLifeAssetID	PortfolioCode	PrincipalBalance	PropertyType	MetLifeInterestTypeCode	MetLifeRatingLoanToValueRatio
LN_0000703144	0BA	31000000.00	OFFICE	FIX	58.45
LN_0000702920	0BA	3729738.55	APART	FIX	68.71
LN_0000701890	0BA	202282.09	INDUS	FIX	32.12
LN_0000703163	0BA	908155.99	RETAI	FIX	58.51
LN_0000702359	0BA	8920737.37	APART	FIX	36.26
LN_0000703102	0BA	**40000000.00**	OFFICE	FIX	51.06
LN_0000520115	0BA	7000000.50	INDUS	FIX	**75.00**
LN_0000703175	0BA	**9999999.00**	RETAI	VAR	9.68
LN_0000520099	0BA	1300000.00	OFFICE	FIX	43.31
LN_0000701895	0BA	244545.86	INDUS	FIX	32.12
LN_0000702521	0BA	45398618.77	OFFICE	FIX	59.79
LN_0000520120	0BA	48500000.30	INDUS	FIX	58.89
LN_0000702686	0BA	15349627.39	RETAI	FIX	64.66
LN_0000702708	0BA	2986402.19	OFFICE	FIX	59.95
LN_0000702755	0BS	21900000.79	Malls	FixedRate	54.62
LN_0000703163	0BS	**555555.00**	Malls	FixedRate	**66.66**
LN_0000702609	0BS	**9999999.00**	Resorts	VariableRate	93.28
LN_0000701746	0BS	1211250.60	Manufacturing	FixedRate	30.97
LN_0000702711	0BS	17200000.30	Workspace	FixedRate	46.36

Demo

GRID DataFrame Framework demo

Demonstrate flow of data & adjustments thru the yellow highlighted components



GRID DataFrame Framework demo

Steps

#	Action	Purpose & outcome
1	Query Asset master dataframe, then export to CSV	Demonstrate ability to query raw ORDZ data via an auto-generated dataframe. Result shows original unaltered values.
2	Query Position + Asset master dataframe, then export to CSV	Demonstrate query of dataframe that combines multiple source dataframes. Result shows original unaltered values.
3	Query aggregated principal balanced dataframe, then export to CSV	Demonstrate query of dataframe that aggregates data from a source dataframes. Result shows original unaltered values.
4	Load adjustments & translations CSV files to ORDZ, run job to propagate to dataframe	Adjustments & Translations CSV file is loaded to the ORDZ, then a job will convert to JSON for consumption by the DFF
5	Query Asset master dataframe, export to CSV	Same as #1, except now see that result has adjustments & translations applied
6	Query Asset master & Position dataframe, export to CSV	Same as #2, except now see that result has adjustments & translations applied
7	Query aggregated principal balance dataframe, export to CSV	Same as #3, except now see that result has adjustments & translations applied