API Data Source

CURRENT STATE REPORT

LAST UPDATE: October 9, 2024

2024

Contents

[1 API Data Source 2](#_Toc179399858)

[1.1 Current state architecture 2](#_Toc179399859)

[1.2 Data Sourcing 2](#_Toc179399860)

[1.2.1 Overview of Data Sources 3](#_Toc179399861)

[1.2.2 Challenges, gaps, and opportunities 3](#_Toc179399862)

[1.3 Data Ingestion 3](#_Toc179399863)

[1.3.1 Technology stack involved and their interactions 3](#_Toc179399864)

[1.3.2 Data Integration Patterns 4](#_Toc179399865)

[1.3.3 Challenges, gaps, and opportunities 4](#_Toc179399866)

[1.4 Data Organization 4](#_Toc179399867)

[1.4.1 Security & Governance Approach 5](#_Toc179399868)

[1.4.2 Challenges, gaps, and opportunities 5](#_Toc179399869)

[1.5 Data Consumption 5](#_Toc179399870)

[1.5.1 Consumption Patterns 5](#_Toc179399871)

[1.5.2 Data Consumers and details on consumption 5](#_Toc179399872)

[1.5.3 Delivery Channels and the performance 5](#_Toc179399873)

[1.5.4 Challenges, gaps, and opportunities 5](#_Toc179399874)

[2 Key Considerations for Future State 5](#_Toc179399875)

[2.1 OA Findings and Understanding 5](#_Toc179399876)

[2.2 Future State Options for the identified Challenges, Gaps, Opportunities 6](#_Toc179399877)

[3 Appendix 6](#_Toc179399878)

# API Data Source

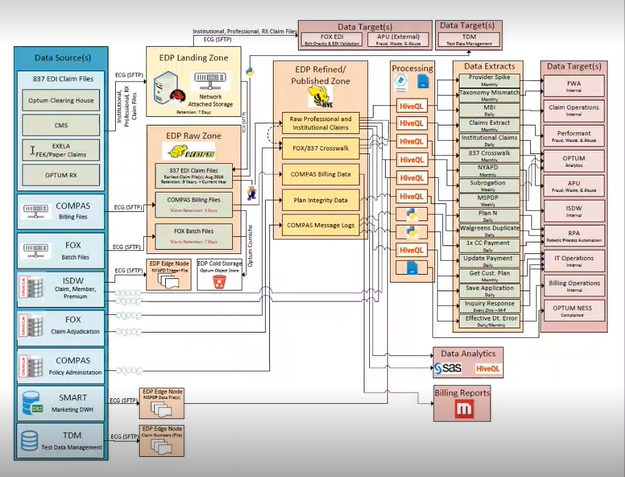
<

API Data source gets data from multiple data sources, such as, Mercury (ServiceNow), OT Datahub, FieldGlass, M&A Masterbrain, SESH (Salesforce), Employee Learning & Development.

>

## Current state architecture

Provide current state architecture.



## Data Sourcing

<

Describe data sources. For example:

* Mercury (ServiceNow)
* OT Datahub
* FieldGlass
* M&A Masterbrain
* SESH (Salesforce)
* Employee Learning & Development

>

### Overview of Data Sources

**Types of Data Sources**

<

* Mercury – ServiceNow
* OT Datahub
* FieldGlass
* M&A Masterbrain – Smartsheet
* SESH – Salesforce
* Employee Learning & Development - Oracle Cloud

>

**Opportunity:**

<

Describe opportunity(s)

>

**Summary of Data Sources**

**Technology Stack**

<

1. Mercury is a homegrown system
2. Alteryx is used to make API calls
3. Microsoft SQL Server is used to store data

>

### Challenges, gaps, and opportunities

<

1. Mercury data load works best in blocks of 3,000 rows
2. The medallion architecture does not have silver layer
3. Overly complex

>

## Data Ingestion

<

After the data is pulled via API calls, the data goes through sanitization and gets stored in the durable storage (Microsoft SQL Server)

>

### Technology stack involved and their interactions

<

1. Alteryx makes API call with specific parameters to the Mercury system
2. Mercury system returns result set in JSON format

>

### Data Integration Patterns

<

The REST API from Alteryx run at 2:00 AM and 5:00 PM CST

Some data is pulled at delta level, but the rest is pulled in its entirety

Then data is stored in its Raw (Bronze) form, after which it goes thru refinement, standardization, and some level of aggregation before it is stored in the ‘Fit for Purpose’ layer

Data is stored in the SQL server

**Note**: It is a process from business systems to payments. It is a business to supplier tool. Mercury system is a central tool. It has 1.2 million records. very compartmentalized. 30 or so API tables.

>

**Data Combination from Various Sources**

**Aggregations and Transformations**

**Loading process to Target**

<

* The final target storage for Mercury data is Microsoft SQL Server

>

**Data Quality Measures**

* RAD team performs validation check to ensure the validity, accuracy, and timeliness of data

### Challenges, gaps, and opportunities

* Pseudo Medallion architecture – missing transformation (silver) layer
* Additional complexity

## Data Organization

**Three layers of data tables:**

**Temporary tables:**

**Database Objects:**

**Data Subject Areas**

**Additional Notes for Reference:**

### Security & Governance Approach

### Challenges, gaps, and opportunities

## Data Consumption

<

* Data Science is a major consumer of this data

>

### Consumption Patterns

### Data Consumers and details on consumption

### Delivery Channels and the performance

<

* Data is delivered via queries

>

### Challenges, gaps, and opportunities

**Gaps:**

**Challenges:**

<

* There is a time lag
* Query response can be slow

>

**Opportunities:**

* + To minimize the lag time
  + Minimize query logic
  + Define proper Medallion schema
  + Empower end consumers

# Key Considerations for Future State

## OA Findings and Understanding

<

* Performance issue
* Data availability lag
* API call return limit 3000 rows
* Very complex
* Lack full Medallion architecture

>

## Future State Options for the identified Challenges, Gaps, Opportunities

<

* Improve Performance
* Minimize Data availability lag
* Overcome return data load size limit of 3000 rows
* Minimize complexity
* Implement full Medallion architecture
* Server end consumers with data products to elevate efficient and simple data access mechanism
* Implement self-discovery and subscription mechanism to empower data consumers

>

# Appendix