**Using Cognite Python SDK to read and write new datasets to CDF - Integration Architecure**

**CoE Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version Control** | | | | |
| Version | Date | Change Descritpion | Responsible | Approver |
| 1.0 | 08.09.2023 | Initial version | Vetle Nevland |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

[1. Document Objective 4](#_Toc41383220)

[2. Problem Descrition 4](#_Toc41383221)

[3. General Graphical Overview 4](#_Toc41383222)

[4. Data Extraction 4](#_Toc41383223)

[5. Transformations 4](#_Toc41383224)

[6. Access Control 4](#_Toc41383225)

[CDF DataSet Configuration 4](#_Toc41383226)

[7. Considerations 4](#_Toc41383227)

[7.1. Premisses 4](#_Toc41383228)

[7.2. Restrictions 4](#_Toc41383229)

[8. Roles and Responsabilties 4](#_Toc41383230)

[9. Applications and Use Cases impacted by New Integration/Change 5](#_Toc41383231)

[10. Infrastructure Impact 5](#_Toc41383232)

# Document Objective

*This document aims at describing the overall architecture of a integration one or more source systems to CDF (RAW or CLEAN). As well as describing the extraction step, it will also describe the following pipeline steps (if applicable), i.e. transformations, contextualisation and any RAW to CLEAN activity.*

A time series dataset containing data points of oil volume percentage in tank is extracted from CDF CLEAN. All transformations are performed using Cognite’s Python SDK. The contextualized data model constitutes the same elements and relationships as the original time series. The new time series is then written back to CDF.

# Problem Descrition

*Here we shall describe what this solution is trying to achieve, source systems, use stories (functional requirements, business rules), CDF resources, etc.*

The goal of the project is to provide a step-by-step tutorial for how to acquire relevant accesses for reading and writing datasets to CDF using Cognite’s Python SDK, as an alternative to CDF’s GUI framework. The project serves as a clear documentation of this process. The only CDF resource type used in this project is Time Series.

The project demonstrate the process by extracting time series data of oil volume percentage in tanks, and transforming the data to calculate daily average drainage rate to help detect leakages.

# General Graphical Overview

*Represent, in a graphical manner, the main components involved in the integration. Here we can use any type of diagram which will help the understanding of the proposed solution: data flow, sequence, etc. Here we should also describe the main interactions (i/o) between the different components.*

The data pipeline comprises a closed loop where data is extracted from Cognite Fusion Prod, transformed with Cognite Functions through Python SDK, written to Cognite Fusion Dev, tested and validated in a testing environment, and finally deployed as a complete data product to Cognite Fusion Prod. The data can then be visualized in, e.g., Grafana Dashboards for further insights.

A diagram of a process

Description automatically generated

# Data Extraction

*Describe the data extraction in a more detailed and technical manner: source systems, servers, technology, data and meta data being transported, destination, RAW (DB and Tables), CLEAN (CDF Resources representation).*

The data of interest is already a compatible CDF resource type (time series) and resides in CDF Clean. The project is thus not dependent on any external source systems or extractors. Metadata attached to the extracted time series also applies for the transformed time series, so no modifications are done to this end.

# Transformations

*Describe all the transformations from RAW to CLEAN in a simple easy to read manner as well as all the technical aspects involved in the transformation, such as type of platform used (GCP, Azure or Cognite), and the different steps (jobs) taken to make it to CLEAN.*

The data has already been transformed to a CDF time series resource type residing in CLEAN. The transformations performed herein are done through Cognite Functions using Cognite’s Python SDK. This includes aggregation, smoothing and calculation of derivative to obtain the daily average drainage rate as a new time series data object. The transformations are as follows:

* Show code snippets of transformations
  + We create a new Time Series CDF data object to contain the transformed time series data: client.time\_series.create(TimeSeries … ).
* client.time\_series.create(TimeSeries(name=ts\_output\_name, external\_id=ts\_output\_name, data\_set\_id=dataset\_id))

The Time Series is assigned to the new dataset created by CDF Operations with id: dataset\_id.

* + Upon retrieval of the time series, we aggregate the signal computing the average value per minute (client.time\_series.data.retrieve(external\_id=cdf\_ext\_id, aggregates=”average”, granularity=”1m”, start=start\_date, end=end\_date))
* client.time\_series.data.retrieve(external\_id=cdf\_ext\_id,
* aggregates="average",
* granularity="1m",
* start=start\_date,
* end=end\_date)
  + Signal is further filtered using Lowess smoothing:
* smooth = lowess(vol\_perc, df['time\_sec'], is\_sorted=True, frac=0.01, it=0)
  + Drainage rate is calculated as the derivative of the smoothed signal:
* df["derivative"] = np.gradient(df['smooth'], df["time\_sec"])
  + Large positive derivatives indicate filling of tanks by humans, while large negative derivatives indicate leakages. We are only interested in the latter, so we omit derivatives of the signal exceeding a given threshold:
* df['derivative\_excl\_filling'] = df["derivative"].apply(lambda x: 0 if x > derivative\_value\_excl or pd.isna(x) else x)
  + The daily average drainage rate in L/min is calculated as
* mean\_drainage\_day = df.groupby('Date')['derivative\_excl\_filling'].mean()\*tank\_volume/100
  + This is converted to a DataFrame with the external ID of our new time series as column name, which is used to populate the Time Series object with the data points
* mean\_df = pd.DataFrame({ts\_output\_name: mean\_drainage\_day})
* ts\_inserted = client.time\_series.data.insert\_dataframe(mean\_df)
  + These transformation steps are encapsulated in a handle function, which is turned into a Cognite Function through the following definition:
* func\_drainage = client.functions.create(
* name="avg-drainage-rate",
* function\_handle=handle
* )

## The transformations are deployed by calling the Cognite Function with necessary input data gathered in data\_dict, and additional information about the function call in func\_info

* call\_func\_drainage = func\_drainage.call(data=data\_dict, function\_call\_info=func\_info)

# Access Control

# To deploy data to the Cognite Fusion Dev evnironment, we need to submit an access request form for Cognite Data Fusion. The form is found here: <https://forms.office.com/Pages/ResponsePage.aspx?id=cEF-O0iDpEq_rgaj4YZ0aUVYsXTN0c9Dil0iHGZgj0lUOTBXVFlSWDlMUFk1WUNBS1lKWjZKWko2TyQlQCN0PWcu>. The following describes how to fill out this form in order to get the necessary access rights for developing our dataset.

# In \**Area of the request\** select Cognite Data Fusion

# In \**Category*\* select New access or create new dataset

# In \**New access or create new dataset request*\* select Create new dataset

# In \**Justification of dataset*\*, mention that calculation of drainage rate will help detecting leakages

# NOE MER

CDF Dataset Configuration:

* There are no specific access restrictions on the produced dataset. General access permissions apply.

**Access groups:**

**DEV/TEST/PROD**

**Group:** UNKNOWN

**Group capabilities**

* *asset:read* and *asset:write,* scoped to the data set.

# 

# Considerations

## Premisses

## Restrictions

# Roles and Responsabilties

# Applications and Use Cases impacted by New Integration/Change

|  |  |  |
| --- | --- | --- |
| **Application** | **Type of Impact (Dev or Test)** | **Responsbile** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Infrastructure Impact

*Here we should define if we have any need of infrastructure involvement (servers, firewalls,etc)*