



*Azure Function Code Explanation*

*For*

FOCUS-BI – Cylindrical Oil Feed Rate Report (File Fetching from Database)

*Author: MOL-IT AID*

*Date Created: 22-Nov-2023*

*Last Revision:*

*Revised By:*

**Guidelines**

**Table of Contents**

[*Overview:* 3](#_Toc151547500)

[*Folder Name: CO\_File\_Fetching\_Activity* 4](#_Toc151547501)

[*File Name: az\_datalake\_activity.py* 7](#_Toc151547502)

# Overview:

This documentation provides an overview of each file's role within the Azure Durable Functions application, from activity processing to orchestrator coordination and blob-trigger-based execution.

Our system utilizes Azure Blob Storage to manage incoming data files. These files are subject to a structured process triggered by a blob update event, invoking a specific Azure Function. This Azure Function is designed to process the received files, make necessary updates, and then replace them in the storage.

The workflow can be outlined as follows:

* Trigger: A blob-triggered Azure Function is initiated upon the update of a designated file, referred to as the "control\_file.txt". It is to be placed on “molpearlqrt” container in a folder named “Online\_ABLOG\_Control\_File”.
* Processing: The Azure Function is responsible for processing the files, performing essential operations, and subsequently updating other files as required.
* Archiving: After successful processing, the originally received files are systematically copied to an archive folder for historical record-keeping.
* Deletion: To maintain an organized storage environment, the processed files are removed from their original location in the source directory.
* Logging: In addition to file processing, log files are generated and uploaded to retain a comprehensive record of the function's execution and data manipulation.

This workflow ensures the efficient management of incoming data, accurate processing, and proper archival, while also maintaining a detailed log of all activities performed by the Azure Function.

Files Description:

1. Incremental\_Data:

* Purpose: This daily-updated file contains incremental data, essential for our processing tasks. Its content evolves daily, necessitating a daily archival process to preserve historical records. It’s data is fetched using SQL Queries and extracted from Focus Database.

# Folder Name: CO\_File\_Fetching\_Activity

Description:

The provided explanation is of an Azure Durable Function designed to fetch and process data from a MySQL database, and subsequently upload the processed data to Azure Blob Storage. The main components of the code include establishing an SSH connection to a remote server, querying a MySQL database, processing the retrieved data using Pandas, creating a CSV file, and uploading the file to Azure Blob Storage. The code is structured as an activity function within the Azure Durable Functions framework and incorporates error handling, logging, and cleanup procedures. The overall goal of this code is to automate the extraction, transformation, and loading (ETL) of data from a database to Azure Blob Storage in a reliable and scalable manner:

1. Importing Libraries and Modules:

A screenshot of a computer

Description automatically generated

The code begins by importing necessary libraries and modules for working with Azure Durable Functions, data processing, and interacting with Azure services.

1. Defining the Activity Function

A black screen with white text

Description automatically generated.

The main function is the entry point of the activity function. It takes a name parameter and returns a string.

1. Initializing Variables and Logging:

A screen shot of a computer

Description automatically generated

Initialization of variables, such as **today\_date**, **blob\_path**, and **destination\_container\_name**. Logging statements are used to provide information about the function's progress.

1. Setting up DataLake Utility File and SSH Key:

A screen shot of a computer screen

Description automatically generated

An instance of azd.DataLakeUtility is created, and the path to an SSH key file is defined. The content of the SSH key file is then read.

1. Establishing SSH Connection:

A screen shot of a computer

Description automatically generated

Paramiko is used to create an SSH client, and the server's host key policy is set. The SSH connection parameters are read from environment variables.

1. Database Connection and Query Execution

A screen shot of a computer

Description automatically generated

An SSH tunnel is established, and a connection to the MySQL database is created using the **pymysql** library. SQL queries are then executed to retrieve specific data from the database.

1. Data Processing with Pandas:

A screen shot of a computer

Description automatically generated

The retrieved data is processed using Pandas. Column names are extracted from the cursor description, and the data is fetched into a Pandas DataFrame.

1. CSV File Creation and Upload to Azure Blob Storage:

A computer screen shot of a computer code

Description automatically generated

The DataFrame is converted to a CSV file, and the file is uploaded to Azure Blob Storage. The blob name is constructed based on yesterday's date.

1. Closing Connection and Cleanup:

A screen shot of a computer

Description automatically generated

The database connection is closed, and temporary directories are deleted as part of cleanup.

1. Exception Handling:

A black and grey rectangular object

Description automatically generated

In case of any exceptions during the execution, an error message is logged, and the function returns the error message.

1. **Function Completion:**

****

If the function completes successfully, it returns a greeting message.

# File Name: az\_datalake\_activity.py

Initialization (\_\_init\_\_): This function initializes the DataLakeUtility object. It sets attributes like guid (a unique identifier for the instance), root\_folder (the temporary directory where downloaded files will be stored), and connection\_string (the connection string to access Azure Blob Storage).

Function Name: connect\_azure

* This method establishes a connection to Azure Blob Storage using the connection string. It returns a BlobServiceClient object that allows you to interact with the Azure Blob service.

Function Name: download\_blob

* This function downloads a specific blob from Azure Blob Storage to a local temporary directory. It takes the container name (contr), blob path (blob\_path), and the file name (file\_name) to save the downloaded file in the local directory. It returns a dictionary with information about the download operation, including the local file's absolute path.

Function Name: upload\_blob

* This method uploads a local file from the temporary directory to Azure Blob Storage. It takes the container name (contr), blob path (blob\_path), the local file's absolute path (temp\_file\_abs\_path), and the desired blob name (blob\_name). It uploads the file and returns a dictionary with information about the upload operation.

Function Name: upload\_log

* This function is specifically designed for uploading log data to Azure Blob Storage. It takes the log data string, blob name, blob path, and the destination container name. It uploads the log data to the specified location and returns a message indicating a successful upload.

**Function Name: delete\_dir**

* This function deletes the temporary directory created for the instance. If the directory exists, it is removed. It returns a dictionary with information about the deletion, including the directory location.

Function Name: copy\_and\_delete

* This method is designed to copy files from one Azure Blob Storage container to another container and then delete them from the source container. It's used for archiving data. The function performs the following steps:
  + Initializes Azure Blob Service clients for both the source and destination containers.
  + Copies specified files from the source container to the destination folder and marks them for deletion.
  + Deletes the copied files from the source container.
  + Logs messages indicating the successful copy and deletion of files.

Please note that in functions like download\_blob and upload\_blob, error handling and logging are implemented to provide detailed information about the operations. The class provides an interface to interact with Azure Blob Storage for common data transfer and archival tasks, making it a useful utility for managing data in Azure Blob Storage.