**DLPROC Manual**

# Deploying DLPROC

Deploying DLPROC is a one-time deployment of the framework, which can be done either during the data lake setup or after it. The deployment is straightforward and does not require external packages or dependencies since the framework is completely python-based.

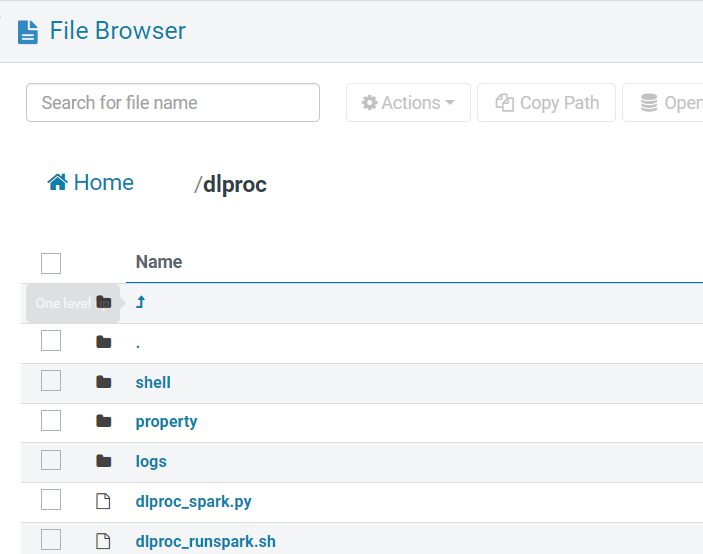
Deployment steps include 3 simple steps as mentioned below:

1. Create required directory structure in storage layer (ADLS/S3/HDFS/..)

|  |
| --- |
| **/dlproc/logs - stores logs of the spark execution**  **/dlproc/property - stores property file for the use-case with Spark-SQLs**  **/dlproc/scripts – stores any scripts / shell files required for the use-case** |

1. Deploying framework specific files in the dlproc directory
   1. dlproc\_runspark.sh in /dlproc
   2. dlproc\_spark.py in /dlproc

The final folder structure would look like this.



1. Creating the metadata database and tables:

Database name: dlproc\_datalake\_monitoring

Tables: dlproc\_audit and dlproc\_spark\_execution\_control

The DDLs are in below file. Please select appropriate DDLs based on metadata database and table type (Hive/Iceberg/PostgresSQL)



# Implementing Spark ETL with DLPROC

Note: below is an example of how to schedule a spark ETL using DLProc.

Note: DLProc is suitable for spark ETLs that are compatible with spark SQL based transformations (mostly BI use-cases). In the case of more complex use-cases which require python code or external python libraries, custom ETL must be built that can then be scheduled using oozie.

## Prepare the Property File

The first step is to create a property file that contains all the Spark SQL queries required for the transformation, listed sequentially. Each step should be uniquely identified, and the final target query step should be clearly marked.

Here’s a redacted version of how the property file will look like.

|  |
| --- |
| **STEP\_1\_SQLQUERY**  SPARK-SQL QUERY 1  **STEP\_1\_TEMPTABLE**  Query1\_Result\_TempName  **STEP\_2\_SQLQUERY**  SPARK-SQL QUERY 2  **STEP\_2\_TEMPTABLE**  Query2\_Result\_TempName  **STEP\_3\_SQLQUERY**  SPARK-SQL QUERY 3  **STEP\_3\_TEMPTABLE**  Query3\_Result\_TempName  **STEP\_4\_TGTQUERY**  INESRT\_QUERY 4  **STEP\_4\_NEXTSTEP**  **END** |

Key Sections of the property file:

**STEP\_X\_SQLQUERY:** This keyword marks the beginning of a SQL query that will be executed. The X represents the sequence number of the step.

**STEP\_X\_TEMPTABLE:** This defines the name of a temporary table where the results of the corresponding SQL query will be stored. The temporary table can be referenced in subsequent queries.

**STEP\_X\_TGTQUERY:** This marks the final target query in the sequence, which typically inserts or updates data in a designated table in the gold layer.

**END:** This keyword indicates the end of the sequence of steps in the property file.

## Place the Property File in storage layer

Place this property file under /dlproc/properties folder in the storage layer.

## Update the Spark Jobs Execution Control Table

This entry controls the execution of the job based on the Ooze schedule.

The entry contains

flowname (unique identifier of the job),

starttime

endtime

Starttime and endtime are used in the property file as the input date range to process data from silver layer.

At the end of the run, an update is expected in the property file that will update the starttime and endtime of this entry to prepare it for the next scheduled run.

lastrun

lastrunstatus

At the end of the run, an update is expected in the property file that will update the lastrun time and status of the flow’s entry for audit purpose.

**Example:** insert into datalake\_monitoring.sparkjobs\_execution\_control values ('gold\_assuranceanalytics\_faultrefs','2024-03-04 00:00:00','2024-03-11 00:00:00',now(),'failed');

## Create & Schedule the Oozie Workflow

In Oozie workflow editor, pull a SHELL action and input the below :

PATH : path of dlproc\_runspark.sh

ARGUMENTS :

Property file name

FILES :

Keytab (placed in hdfs)

## A screenshot of a computer Description automatically generated

Save the workflow with a name

Then, Create the Oozie schedule to schedule the workflow just created.