DFS TD - Application Lab

Specific technical documentation for the Application Lab tests

Summary

Data sources

R aw data files for all labs are extracted from the servers presented in the following spreadsheet: Asset Tag Spreadsheet

The table below presents all the Application tests that are currently included in the Datalake, the source of data and the raw data files format for each test:

Test	Data Sources	File format
ODR (Oscillating Disc Rheometer)	Raw data files ELN data	.exp .xml
DSAS (Dynamic Strain Amplitude Sweep)	Raw data files ELN data	.csv .xml
DSAS_U (Dynamic Strain Amplitude Sweep Uncured) Programs : P05, P28 and P38	Raw data files ELN data	.csv .xml
Tensile Properties	Raw data files ELN data	.csv .xml
Temperature Sweep	Raw data files ELN data	.asc, .csv .xml
DIN Abrasion Resistance	ELN data	.xml
Filler Dispersion	ELN data	.xml
Hardness Shore A	ELN data	.xml
Lambourn	ELN data	.xml
Mixing Process	ELN data	.xml
Mooney Viscosity	ELN data	.xml
Abrasion Loss and Density	ELN data	.xml
Rebound Resilience	ELN data	.xml
Tear Resistance	ELN data	.xml
DSA (Dynamic Strain Amplitude)	ELN data	.xml

Talend jobs

Data is processed using Talend and Python. Talend jobs regarding Application Lab can be found in the Talend project **Rnl_Silica_Application** .

Job **F000_Orch_Flow_Application** is the orchestrator and will call successively all the other jobs a determined order.

Python scripts

A copy of the Python scripts for downloading, parsing and computing new fields and columns for the Application Lab raw data and ELN data can be found here:

However it's recommended to get the last version of the scripts available in the production path: D:\DATA\PROD\RnI\Silica\Permanent. For the moment, **git is not in place for this project.**

ELN Data jobs

- **F001_ELN_Application_Get_XML_ELN**
  - extracts spreadsheets from ELN that were added or modified in the last 24h as .zip archives. The .xml files are extracted by using four successive API calls. For details on this process see: https://drive.google.com/drive/folders/1-rRqOn4CfA7VOpVcHESJTraBg9mrIViw?usp=drive_link
  - The ELN folders are defined in the file: ...\DATA\[ENV]\RnI\Silica\Input\Application\ELNsubFolder.csv

> The Po2 project will need to change only the defined folders in the file above in order to get the right ELN data

- F002_ELN_Application_Raw_xml_Upload_to_CloudStorage
  - uploads the raw .xml files into a Google Storage bucket
- F003_ELN_Application_Unzip_Files
  - unzips all the .zip archives and extracts the .xml files
- F004 to F021
  - These jobs extract all the tables and fields from the .xml files for each test. The following Python scripts are integrated into the parsing jobs and they extract useful information and put it in an exploitable form before loading it in BigQuery:
    - parse_StrudyDefinition_ELN.py (F004)
    - parse_DIN_AbrasionResistance_ELN.py (F005)
    - parse_DSAS_ELN.py (F006)
    - parse_FillerDispersion_ELN.py (F007)
    - parse_HardnessShoreA_ELN.py (F008)
    - parse_Lambourn_ELN.py (F009)
    - parse_Mixing_ELN.py (F010)
    - parse_MooneyViscosity_ELN.py (F011)
    - parse_ODR_ELN.py (F012)
    - parse_TemperatureSweep_ELN.py (F013)
    - parse_TensileProperties_ELN.py (F014)
    - parse_AbrasionLossDensity_ELN.py (F015)
    - parse_ReboundResilience_ELN.py (F016)
    - parse_TearResistance_ELN.py (F017)
    - parse_DSA_ELN.py (F018)
    - parse_DSAS_U_P05_ELN.py (F019)
    - parse_DSAS_U_P28_ELN.py (F020)
    - parse_DSAS_U_P38_ELN.py (F021)
- F020_ELN_Data_Application_Upload_to_BQ
  - uploads the following tables into the BigQuery dataset *ELN_data_application_mig*:
    - Abrasion_Loss_and_Density_Synthetic_results
    - All_tests_Experimental_condition_range_value
    - All_tests_Experimental_condition_single_value
    - All_tests_Experimental_condition_text
    - All_tests_Experimental_details
    - DIN_abrasion_resistance_Synthetic_results
    - DSA_Synthetic_results
    - Filler_dispersion_Synthetic_results
    - General_Information_Application
    - Hardness_Shore_A_Synthetic_results
    - Lambourn_Synthetic_results
    - Mixing_Raw_data
    - Mixing_Synthetic_results
    - Mooney_viscosity_Synthetic_results
    - Rebound_Resilience_Synthetic_results
    - Study_definition_Mix_and_formulation
    - Study_definition_Operating_methods
    - Study_definition_Phase_list

        - Study_definition_Study_details
        - StudyTestMix_Mapping
        - Tear_Resistance_Synthetic_results

- **F021_IDBS_Application_Extract_Data**
    - The users of the Tableau application of the datalake inherit the rights to access data from the ELN (i.e., they will have the right to access in Tableau only the data corresponding to the spreadsheets that they are allowed to access in ELN. Thus, the job F021 is meant to extract from the Oracle DB the permissions for each ELN Silica User.

- **F022_IDBS_Application_Prepare_and_Upload_to_BQ**
    - For each user, lists the Unique IDs of the spreadsheets that he is allowed to access and uploads these information to BigQuery, in the dataset access_rights_mig, the table idbs_access_rights_application.

Raw Data jobs

- R001_Download_Application - Downloads raw data files from the lab servers.

For the tests for which raw data files are available (i.e., DSAS, DSAS_U, ODR, Tensile Properties, Temperature Sweep), the table raw_data_mapping previously extracted from the ELN .xml files contains, for each test_id, the link to the raw data file that needs to be extracted. For each test, the raw data file extraction is made via the Python scripts incorporated in this job (R001):

- download_CP_ODR.py
- download_DP_DSA_Sweep
- download_GP_DSAS_U
- download_PP_TP
- download_temp_sweep

In order to ensure a maximum of flexibility for the user, raw data files are extracted and processed each day. Therefore, the history of raw data files is not stored from one day to the next, every raw data file is extracted every day.

One advantage of this approach is that the user has the possibility to modify or to change the raw data files and to see these changements reflected in the datalake. Nevertheless, this implies that the link to the raw data file written in ELN must always be up to date and it needs to allow the extraction of the file. This condition is fulfilled if (1) the lab server and (2) the arborescence leading to the raw data file do not change.

It is possible that lab servers change due to the replacement of the lab machines. As mentioned above, this has consequences on the extraction of the raw data files because the link to the raw data file written in ELN becomes obsolete. The python scripts that extract the raw data files are able to adapt and to handle the changes of the lab machines without the intervention of the user (No need to modify the link to the file that was written in ELN). Nevertheless, this solution is viable only if the following conditions are fulfilled:

1. The name of the root folder containing the raw data needs to stay the same (i.e. Data lake, or any other name that was initially established for each test).
2. From this level of the arborescence (Data lake\...), the path to the raw data file needs to stay unchanged
3. Every change of lab machine needs to be reported as soon as possible to our team. The new name of the server needs to be updated in the variable contexts of the Talend project:

```
-- SOURCE DIRs FOR APPLICATION LAB
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_TempSweep','\\FRCOL-LABPC-BACKUP\LABO\FRCOL31531',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_TempSweep_bis','\\FRCOL-LABPC-BACKUP\LABO\FRCOL28477',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_GP_DSAS_U','\\FRCOL-LABPC-BACKUP\LABO\FRCOL33595',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_CP_ODR','\\FRCOL-LABPC-BACKUP\LABO\FRCOL33291',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_CP_ODR_bis','\\FRCOL-LABPC-BACKUP\LABO\FRCOL33265',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_PP_TP','\\FRCOL-LABPC-BACKUP\LABO\W-512283\Output',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_DSAS','\\FRCOL-LABPC-BACKUP\LABO\FRCOL28477',@project,'',@d
Exec ##upsertParam @env,'1_DIR_RNI_SILICA_Application_DSAS_bis','\\FRCOL-LABPC-BACKUP\LABO\FRCOL31531',@project,'',@d
```

- **R002_Raw_Data_Application_Upload_to_CloudStorage**
    - uploads the raw data files into a Google Storage bucket

- R003_Prepare_Application
    - Extracts useful data from each raw data file and generates computed fields and resume tables for each test. The details of the computations for each test and the python scripts that are used are detailed in the following sections

- R004_Raw_Data_Application_Upload_to_BQ
    - uploads the following tables, generated in the previous step, into the BigQuery dataset *raw_data_application* :
        - ApplicationCuringProperties_ODR
        - ApplicationDynamicalProperties_DSAS
        - ApplicationDynamicalProperties_TemperatureSweep
        - ApplicationGreenProperties_DSAS_U
        - ApplicationPhysicalProperties_PP_TP
        - ComputedApplicationCuringProperties_ODR

- ComputedApplicationDynamicalProperties_DSAS
- ComputedApplicationDynamicalProperties_TemperatureSweep
- ComputedApplicationGreenProperties_DSAS_U
- ComputedApplicationPhysicalProperties_PP_TP
- ComputedApplicationPhysicalProperties_PP_TP_Aggregated

Application Lab Tests

Curing Properties – Oscillating Disc Rheometer (ODR)

The Python script **download_CP_ODR.py** extracts the raw data files listed in the ELN table raw_data_mapping from the lab server.

For each raw data file, the Python script **parse_CP-ODR.py** creates a .csv file containing the following columns extracted from the raw data file:

- test_id
- time_sec
- time_min
- s_prim
- temperature
- speed

For each file generated in the previous step, the python script **compute_CP-ODR.py** creates a .csv file containing the following values computed from raw data:

- **min_torque**

Minimum of “s_prim”

- **max_torque**

Maximum of “s_prim”

- **final_torque**

Last value of “s_prim”

- **delta_torque**

max_torque – min_torque

- **TS2**

Value of “time_min” corresponding to min_torque + 2

- **T90**

Value of “time_min” corresponding to delta_torque * 90% + min_torque

- **T98**

Value of “time_min” corresponding to delta_torque * 98% + min_torque

- **T98_50**

T98 * 1.5

- **reversion_index**

Compute the slope of “s_prim” starting with time T90.

reversion_index = slope if slope < 0

reversion_index = 0 if slope > 0

- **DT_mT**

delta_torque / min_torque

- **curing_speed**

Value of “speed” corresponding to T50

Dynamical Properties – Dynamic Strain Amplitude Sweep (DSAS)

The Python script **download_DP_DSA_Sweep.py** extracts the raw data files listed in the ELN table raw_data_mapping from the lab server.

For each raw data file, the Python script **parse_DP_DSA_Sweep** creates a .csv file with the following columns extracted from the raw data files:

- test_id
- dsa
- g_star
- g_prime
- g_second
- tan_delta

For each test_id, the Python script **compute_DP_DSA_Sweep.py** creates a .csv file with the following values computed from raw data:

- **g_prime_0_first**

First value of “g_prim”

- **g_prime_0_second**

Last value of “g_prim”

- **g_prime_50**

Value of “g_prim” corresponding to “dsa” = 0.5

- **g_prime_0_second_50**

g_prime_50 - g_prime_0_second

- **g_second_max**

Maximum of “g_second” for the descent

- **strain_max_g**

Value of “dsa” corresponding to g_second_max

- **tan_d_max**

Maximum of “tan_delta” for the descent

- **tan_d_10**

Value of “tan_d” at 10% of “dsa” for the descent

- **tan_d_01**

Value of “tan_d” at 0.1% of “dsa” for the descent

- **g_star_12**

Value of “g_star” at 12% of “dsa” for the descent

Dynamical Properties – Temperature Sweep

The Python script **download_Temp_Sweep.py** extracts the raw data files listed in the ELN table raw_data_mapping from the lab server.

For each test_id, the Python script **parse_DP_Temp_Sweep.py** creates a .csv file with the following columns extracted from the raw data files:

- test_id
- temperature
- e_star
- e_prime
- e_second
- tan_delta

For each test_id, the Python script **parse_DP_Temp_Sweep.py** creates a .csv file with the following values computed from raw data:

- **max_tan_delta**

Maximum of “tan_delta”

- **temp_at_max_tan_delta**

Value of “temperature” corresponding to max_tan_delta

- **max_esec**

Maximum of “e_second”

- **temp_at_max_esec**

Value of “temperature” corresponding to max_esec

- **estar_min_70**

Linear regression to infer the value of “e_star” corresponding to “temperature” = -70

- **estar_min_45**

Linear regression to infer the value of “e_star” corresponding to “temperature” = -45

- **estar_min_25**

Linear regression to infer the value of “e_star” corresponding to “temperature” = -25

- **tan_delta_min_25**

Linear regression to infer the value of “tan_delta” corresponding to “temperature” = -25

- **tan_delta_0**

Linear regression to infer the value of “tan_delta” corresponding to “temperature” = 0

- **eprime_0**

Linear regression to infer the value of “e_prime” corresponding to “temperature” = 0

- **esec_0**

Linear regression to infer the value of “e_second” corresponding to “temperature” = 0

- **tan_delta_40**

Linear regression to infer the value of “tan_delta” corresponding to “temperature” = 40

- **tan_delta_60**

Linear regression to infer the value of “tan_delta” corresponding to “temperature” = 60

- **tan_delta_80**

Linear regression to infer the value of “tan_delta” corresponding to “temperature” = 80

- **estar_60**

Linear regression to infer the value of “e_star” corresponding to “temperature” = 60

- **eprime_40**

Linear regression to infer the value of “e_prime” corresponding to “temperature” = 40

- **tan_delta_80_100**

tan_delta_100 / tan_delta_80

- **estar_min_100_min_80**

estar_min_100 / estar_min_80

Green Properties – Dynamic Strain Amplitude Sweep Uncured (DSAS_U)

The Python script **download_GP_DSAS_U.py** extracts the raw data files listed in the ELN table raw_data_mapping from the lab server.

Raw data files for DSAS_U are extracted only for the P38 program. The python scripts were designed to handle P5 and P28 programs also, but these parts of the code need to be deleted once we will be sure that they are no longer needed.

For each raw data file, the Python script **parse_GP_DSAS_U.py** creates a .csv file with the following columns extracted from raw data files:

- test_id
- time
- set_strain_perc
- g_prim
- g_sec
- tan_delta

For each test_id, the Python script **compute_GP_DSAS_U** (P05, P28, P38) creates a .csv file with the following values computed from raw data:

- **g_prim_091**

Value of “g_prim” corresponding to the value of “set_strain_perc” that is closest to 0.91%

- **g_prim_14**

Value of “g_prim” corresponding to the value of “set_strain_perc” that is closest to 14%

- **g_prim_50**

Value of “g_prim” corresponding to the value of “set_strain_perc” that is closest to 50%

- **delta_g_prim**

g_prim_091 - g_prim_50

- **tand_091**

Value of “tan_delta” corresponding to the value of “set_strain_perc” that is closest to 0.91%

- **tand_50**

Value of “tan_delta” corresponding to the value of “set_strain_perc” that is closest to 50%

- **g_sec_091**

Value of “g_sec” corresponding to the value of “set_strain_perc” that is closest to 0.91%

- **g_sec_14**

Value of “g_sec” corresponding to the value of “set_strain_perc” that is closest to 14%

- **g_sec_50**

Value of “g_sec” corresponding to the value of “set_strain_perc” that is closest to 50%

- **delta_g_sec**

g_sec_091 - g_sec_50

Physical Properties – Tensile Properties

From ELN data, the Python script **download_TensileProperties.py** extracts de links to the raw data files that need to be extracted from the server and downloads these files.

Raw data files are in .csv format and they are organized by study ID. The raw data files corresponding to one study ID should be put in a folder named with the sample ID only.

Each folder corresponding to one study ID contains one or several sub-folders. Each sub-folder contains raw data files corresponding to one single test_id and it’s name should be composed by the test_id followed by the string “.is_tens_RawData” (ex., the sub-folder “19cc355-01.is_tens_RawData”).

Along with the sub-folders, the folder corresponding to the study_id contains one .csv file corresponding to each sub-folder (ex., for the sub-folder “19cc355-01.is_tens_RawData”, we will have the .csv file “19cc355.is_tens.csv”). These .csv files contain the list of specimens for which data will be extracted.

For each test_id and for each specimen, the Python script **parse_TensileProperties.py** creates a .csv file with the following columns extracted or computed from raw data files **:**

- **test_id**
- specimen
- abcissa - list of values (0, 10, 20, 30…)
- constraint_traction
- secant_modulus

(deformation_traction * constraint_traction) / 100

For each test_id, the **average specimen** is also computed.

For the average specimen of each test_id, the following columns are computed:

- **sm_avg**

Average of secant_modulus of all specimens / “abcissa” * 100

- **J**

Average of secant_modulus of all specimens * (“abcissa” / 100 + 1)

- **K**

“J” / “abcissa” * 100

The Python script **compute_TensileProperties.py** creates a .csv file with the following computed values for each test_id and for each specimen:

- **m10**

Value of “constrant_traction” corresponding to “abcissa” = 10

- **m100**

Value of “constrant_traction” corresponding to “abcissa” = 100

- **m200**

Value of “constrant_traction” corresponding to “abcissa” = 200

- **m300**

Value of “constrant_traction” corresponding to “abcissa” = 300

- **di_m10_m100**

m10 / m100

- **ir_m200_m100**

m200 / m100

- **ir_m300_m100**

m300 / m100

- **ts**

Maximum of “constraint_traction”

- **eb**

Value of abcissa corresponding to ts

- **ebts**

ts * eb