**Knowledge Transfer Document: Excel Utility for Robot Framework with Python**

**Objective**

This document provides detailed information about an Excel utility created to simplify data-driven testing in the Robot Framework with Python. The utility allows users to read data from an Excel file and retrieve values dynamically, enhancing the flexibility and reusability of test scripts.

**Features of the Excel Utility**

* Enables seamless integration of Excel data into Robot Framework scripts.
* Provides dynamic data retrieval by allowing filtering using specific parameters.
* Supports data-driven test cases by reading Excel data efficiently.

**Utility Methods**

**1. get\_workbook\_data**

**Purpose:**  
Fetches specific data from an Excel sheet based on a given scenario name, target column, and other parameters.

**Parameters:**

* **scenario\_name**: A unique identifier or scenario keyword to filter the data.
* **path**: The relative or absolute path to the Excel file.
* **sheetname**: Name of the sheet to read data from.
* **target\_column**: The column whose data needs to be extracted.

**Usage Example:**

python

Copy code

data = get\_workbook\_data(

scenario\_name="LoginTest",

path="testData.xlsx",

sheetname="TestData",

target\_column="ExpectedOutput"

)

print(data) # Outputs the filtered data for "LoginTest"

**2. get\_value\_by\_header**

**Purpose:**  
Retrieves a specific cell value by using the column name as a header, providing flexibility to work with row indices and column headers.

**Parameters:**

* **path**: The relative or absolute path to the Excel file.
* **sheetname**: Name of the sheet to read data from.
* **row\_index**: The row index (zero-based) from which the value needs to be fetched.
* **column\_name**: The header of the column to locate the data.

**Usage Example:**

python

Copy code

value = get\_value\_by\_header(

path="testData.xlsx",

sheetname="TestData",

row\_index=3,

column\_name="Password"

)

print(value) # Outputs the value in the "Password" column for row 3

**Implementation Details**

**Setup Steps**

1. **Install Required Libraries:**  
   Ensure you have the following Python libraries installed:
   * openpyxl (for reading Excel files)
   * Any other relevant dependencies.

Use the command:

bash

Copy code

pip install openpyxl

1. **Folder Structure:**  
   The Excel utility is designed to fit seamlessly into your Robot Framework project structure. Here’s an example:

Copy code

Project/

├── RobotPython/

│ ├── Utilities/

│ │ └── excel\_utility.py

│ ├── Tests/

│ │ └── test\_cases.robot

│ └── testData.xlsx

**Code for Methods**

**get\_workbook\_data:**

python

Copy code

import openpyxl

def get\_workbook\_data(scenario\_name, path, sheetname, target\_column):

workbook = openpyxl.load\_workbook(path)

sheet = workbook[sheetname]

data = {}

for row in sheet.iter\_rows(values\_only=True):

if row[0] == scenario\_name: # Assuming first column contains the scenario name

data = {sheet.cell(1, col + 1).value: row[col] for col in range(sheet.max\_column)}

break

return data.get(target\_column)

**get\_value\_by\_header:**

python

Copy code

import openpyxl

def get\_value\_by\_header(path, sheetname, row\_index, column\_name):

workbook = openpyxl.load\_workbook(path)

sheet = workbook[sheetname]

headers = {cell.value: idx for idx, cell in enumerate(sheet[1])}

col\_index = headers.get(column\_name)

if col\_index is not None:

return sheet.cell(row=row\_index + 1, column=col\_index + 1).value

return None

**Integration with Robot Framework**

To integrate the utility into your Robot Framework scripts:

1. Add a Library Import:

robot

Copy code

Library ../Utilities/excel\_utility.py

1. Use Keywords in Tests:

robot

Copy code

\*\*\* Test Cases \*\*\*

Fetch Scenario Data

${data}= Call Method get\_workbook\_data LoginTest testData.xlsx TestData ExpectedOutput

Log ${data}

Fetch Specific Cell Value

${value}= Call Method get\_value\_by\_header testData.xlsx TestData 3 Password

Log ${value}

**Best Practices**

1. Ensure Excel files are well-structured, with headers in the first row.
2. Use relative paths for portability across environments.
3. Validate the presence of headers and data before processing to avoid runtime errors.

**Advantages**

* Centralized data management using Excel.
* Minimizes hardcoding of test data in scripts.
* Improves script reusability and maintainability.

**Conclusion**

This utility serves as a robust solution for managing test data in Robot Framework, leveraging Python’s capabilities for data extraction. Its dynamic and flexible design ensures ease of use and adaptability for various testing scenarios.