Work Instructions

Number: **002**

Name: **Raw data preparation**

Title: Preparation of files of raw data for import into the preparation and profiling tool

Status: Approved

Version: 1.0

Date: 2023-06-30

Author: Michael Rubinigg

This document is intended for the regulation of processes within the EU Pollinator Hub. It is published under the CC BY 4.0 licence. You may distribute, transform, adapt and build upon the material in any medium or format, including for commercial purposes, as long as you give credit to the author.

Table of contents

1 Document history 3
1.1 Release 3
1.2 Change history 3
2 Abbreviations 4
3 Introduction 5
4 Validity 6
4.1 Area 6
4.2 Time 6
4.3 Related Documents 6
5 Procedures 7
5.1 Raw data preparation with Excel 7
5.2 Raw data files with tab separated values 9
5.3 Raw data files in JSON format 10
5.4 Raw data files in HTML format 11
5.5 Raw data files in separated text files 12
6 References 13
7 Annexes 14
7.1 Python scripts 14

1 Document history

1.1 Release

1.1.1 Drafted

Submitted on 2022-10-14 by Michael Rubinigg, quality manager.

1.1.2 Reviewed

Reviewed on 2023-06-30 by Noa Simon Delso, general manager.

1.1.3 Approved

Approved on 2023-06-30 by Michael Rubinigg, quality manager.

1.2 Change history

n.a.

2 Abbreviations

IDE	Integrated development environment
n.a.	not available
WI	Work Instructions

3 Introduction

This document contains instructions for the preparation of files of raw data for import into the preparation and profiling tool.

4 Validity

4.1 Area

The of validity of this document is limited to the preparation of files of raw data for import into the preparation and profiling tool.

4.2 Time

The period of validity of this WI begins on 2023-06-30 and ends with its revocation.

4.3 Related Documents

n.a.

5 Procedures

5.1 Raw data preparation with Excel

This procedure is followed if raw data comes in Excel spreadsheets or formats compatible with Excel.

5.1.1 Preparatory file

5.1.1.1 Creation of new spreadsheet

- Create a preparatory file with the following name: [Short title of the project]_***PREP***_[initials of the operator]_[date of file creation using the format YYMMDD].[File Extension] using Excel (Microsoft Corporation)
- Save the preparatory file in the directory ***01 Raw Data*** of the dataset.

5.1.1.2 Configuration of the new spreadsheet

- Set the decimal separator required in phpMyAdmin.
  - Go to ***File*** > ***Options*** > ***Advanced***.
  - Under ***Editing options***, clear the ***Use system separators*** check box.
  - Under Decimal separator enter: . (dot)

5.1.1.3 Creation of worksheet

- Create a worksheet.
  - The name of worksheet and the name of the corresponding table in database of the profiling tool to which the data will be imported must be the same.
- Repeat all previous steps of paragraph 5.1.1.2 for all tables contained in the dataset.

5.1.2 Configuration of worksheet

5.1.2.1 Definition of header row

- Prepare a single header row.
  - Assign a unique name to a column which corresponds to the name of the column used in the profiling tool to which the data will be imported.
  - Repeat the previous step for all columns.
- Set a data filter on all header fields.

5.1.2.2 Definition of data types

- Define the data type of a column in the preparatory file according to the requirements for the import into the profiling tool to which the data will be imported:

- For a column containing alphanumericals use the cell format category “Text”.
- For a column containing numbers
  - Use the cell format category “***Number***”.
  - Select the maximum number of decimal places occurring in the raw data.
  - Clear the ***Use 1000 Separator*** Checkbox.
- For a column containing date use the format YYYY-MM-DD.
- For a column containing date and time use the format YYYY-MM-DD hh:mm:ss.

- Repeat all previous steps of paragraph 5.1.2.2 for all columns.

5.1.2.3 Transfer of data

- Transfer the values of the raw data to the corresponding worksheet in the preparatory file.
- Repeat the previous step for all tables contained in the raw data files.

5.1.2.4 Define missing values

- Open a column header.
- Clear the ***(Select all)*** checkbox.
- Check all checkboxes of missing values that should be replaced in the profiling tool with NULL.
- Replace the fields of the column in all displayed records with NULL.
- Check the ***(Select all)*** checkbox.
- Repeat all previous steps of paragraph 5.1.2.4 for all columns contained in the worksheet.

5.1.3 Export of raw data in preparatory file

5.1.3.1 Export of raw data

- Use the ‘Save as’ function of Excel to export each worksheet of the preparatory file.
  - Select the directory ***01 Raw Data*** of the dataset as target folder.
  - Use the following name as file name: [Name of the worksheet]_[Short title of the project]_***PREP***_[initials of the operator]_[date of creation using the format YYMMDD].[File Extension] using Excel (Microsoft Corporation)
  - As file type select *CSV UTF-8 (comma delimited) (*.csv)*
  - Save the file in the selected directory.
- Repeat all previous steps of paragraph 5.1.3.1 for all worksheets contained in the preparatory file.

5.2 Raw data files with tab separated values

This procedure is followed if raw data comes in a text file containing tab separated values.

5.2.1 Command prompt window (Windows)

- Open the command prompt window.
- Enter *py* in the commands line.

5.2.2 Integrated development environment (IDE)

- Start the IDE.
- Proceed according to the workflow of the IDE.

5.2.3 Export of raw data to preparatory file

5.2.3.1 Run the conversion script

- Run the script *ConvertTsv2Csv.py* in section 7.1.1.
- Use the following name as output file name: [Name of the input file]_[Short title of the project]_***PREP***_[initials of the operator]_[date of creation using the format YYMMDD]**.csv**
- Save the output file in directory ***01 Raw Data*** of the dataset.
- Repeat all previous steps of paragraph 5.2.3.1 for all raw data files.

5.3 Raw data files in JSON format

This procedure is followed if raw data comes in a file in any type of JSON format.

5.3.1 Command prompt window (Windows)

- Open the command prompt window.
- Enter *py* in the commands line.

5.3.2 Integrated development environment (IDE)

- Start the IDE.
- Proceed according to the workflow of the IDE.

5.3.3 Export of raw data to preparatory file

5.3.3.1 Run the conversion script

- Run the script *ConvertJson2Csv.py* in section 7.1.2.
- Use the following name as output file name: [Name of the input file]_[Short title of the project]_***PREP***_[initials of the operator]_[date of creation using the format YYMMDD]**.csv**
- Save the output file in directory ***01 Raw Data*** of the dataset.
- Repeat all previous steps of paragraph 5.3.3.1 for all raw data files.

5.4 Raw data files in HTML format

This procedure is followed if raw data comes in a file in any type of HTML format.

5.4.1 Command prompt window (Windows)

- Open the command prompt window.
- Enter *py* in the commands line.

5.4.2 Integrated development environment (IDE)

- Start the IDE.
- Proceed according to the workflow of the IDE.

5.4.3 Export of raw data to preparatory file

5.4.3.1 Run the conversion script

- Run the script *ConvertHtml2Csv.py* in section 7.1.3
- Use the following name as output file name: [Name of the input file]_[Short title of the project]_***PREP***_[initials of the operator]_[date of creation using the format YYMMDD]**.csv**
- Save the output file in directory ***01 Raw Data*** of the dataset.
- Repeat all previous steps of paragraph 5.4.3.1 for all raw data files.

5.5 Raw data files in separated text files

This procedure is followed if raw data comes in separated text files that have to be merged prior to processing.

5.5.1 Command prompt window (Windows)

- Open the command prompt window.
- Enter *py* in the commands line.

5.5.2 Integrated development environment (IDE)

- Start the IDE.
- Proceed according to the workflow of the IDE.

5.5.3 Export of raw data to preparatory file

5.5.3.1 Run the conversion script

- Run the script *MergeTextFiles.py* in section 7.1.4
- Use the following name as output file name: [Name of the input file]_[Short title of the project]_***PREP***_[initials of the operator]_[date of creation using the format YYMMDD]**.csv**
- Save the output file in directory ***01 Raw Data*** of the dataset.
- Repeat all previous steps of paragraph 5.5.3.1for all raw data files.

6 References

There are no sources in the current document.

7 Annexes

7.1 Python scripts

7.1.1 ConvertTsv2Csv.py

```
# Title: ConvertTsv2Csl.py
# Author: Rubinigg Michael
# Contributor: Rubinigg Michael
# Abstract: Converts tab separated value files of any structure to CSV files based on user input
# Version 1.0
# Release date: 2023-09-12
# Place: Graz, Austria, EU
# Company: dr Mag. rer. nat. Michael Rubinigg
# Programming Language: Python 3.11
# Rights:  Creative Commons Attribution 4.0 (CC-BY)

import tkinter as tk
from tkinter import filedialog
import csv

def convert_tab_to_csv(input_file_path, output_file_path):
    try:
        with open(input_file_path, 'r', newline='', encoding='utf-8') as input_file:
            with open(output_file_path, 'w', newline='', encoding='utf-8') as output_file:
                tab_reader = csv.reader(input_file, delimiter='\t')
                csv_writer = csv.writer(output_file, delimiter=',')

                for row in tab_reader:
                    # Replace missing data with "NULL"
                    cleaned_row = ['NULL' if cell.strip() == '' else cell for cell in row]
                    csv_writer.writerow(cleaned_row)

        print("Conversion completed successfully.")

    except Exception as e:
        print("An error occurred:", str(e))

def select_input_file():
    input_file_path = filedialog.askopenfilename(filetypes=[("Tab-Delimited Files", "*.txt")])
    return input_file_path

def select_output_file():
    output_file_path = filedialog.asksaveasfilename(defaultextension=".csv", filetypes=[("CSV
Files", "*.csv")])
    return output_file_path

def main():
#    tk.Tk().withdraw()  # Hide the main tkinter window
    print("Select the input tab-separated text file.")
    input_file_path = select_input_file()

    if not input_file_path:
        print("No input file selected. Exiting.")
        return

    print("Select the output CSV file.")
    output_file_path = select_output_file()

    if not output_file_path:
        print("No output file selected. Exiting.")
        return

    # Perform the conversion
    convert_tab_to_csv(input_file_path, output_file_path)

if __name__ == "__main__":
    main()
```

7.1.2 ConvertJson2Csv.py

```
# Title: ConvertJson2Csl.py
# Author: Rubinigg Davide
# Contributor: Rubinigg Michael
# Abstract: Converts JSON files of any structure to CSV files based on user input
# Version 1.0
# Release date: 2023-06-19
# Place: Graz, Austria, EU
# Company: dr Mag. rer. nat. Michael Rubinigg
# Programming Language: Python 3.11
# Rights:  Creative Commons Attribution 4.0 (CC-BY)

import json

import pandas as pd
from tkinter import filedialog as fd, simpledialog
import tkinter as tk
from pathlib import Path

def extract_translations(data, path, translations_list):
    if isinstance(data, dict):
        for key, value in data.items():
            new_path = path + [key]
            extract_translations(value, new_path, translations_list)
    elif isinstance(data, str):
        translation = path + [data]
        translations_list.append(translation)
    else:
        translation = path + [data]
        translations_list.append(translation)

def get_string_input():
    # Create the Tkinter window
    window = tk.Tk()
    window.withdraw()

    # Prompt the user for string input
    user_input = simpledialog.askstring("String Input", "Enter a string:")

    # Close the Tkinter window
    window.destroy()

    # Return the user input
    return user_input

def get_data_by_path(data, root_path: str):
    root_path = root_path.split(",")

    root_object = data
    for key in root_path:
        if key in root_object:
            root_object = root_object[key]
        else:
            raise KeyError(f"Key '{key}' not found in the JSON data.")

    return root_object

def extract(file_path: Path, json_path: str, output_path: Path):
    with open(file_path, encoding='utf-8') as inputfile:
        data = json.loads(inputfile.read())
        data = get_data_by_path(data, json_path)
        translations_list = []
        extract_translations(data, [], translations_list)

        data_frame = pd.DataFrame(translations_list)
        data_frame.to_csv(output_path, index=False)

def main():
    input_path = Path(fd.askopenfilename())
    root_path = get_string_input()
    output_path = Path(fd.asksaveasfilename(defaultextension=".csv", filetypes=[("CSV Files",
"*.csv")]))
    extract(input_path, root_path, output_path)
```

```
if __name__ == '__main__':
    main()
```

7.1.3 ConvertHtml2Csv.py

```
# Title: ConvertHtml2Csv.py
# Author: Rubinigg Michael
# Contributor: Rubinigg Michael
# Abstract: Converts an html table to CSV files based on user input
# Version 1.0
# Release date: 2023-09-13
# Place: Graz, Austria, EU
# Company: dr Mag. rer. nat. Michael Rubinigg
# Programming Language: Python 3.11
# Rights:  Creative Commons Attribution 4.0 (CC-BY)

import csv
from bs4 import BeautifulSoup
from tkinter import Tk, filedialog

def html_to_csv(input_file, output_file, table_class):
    with open(input_file, 'r', encoding='utf-8') as f:
        soup = BeautifulSoup(f, 'html.parser')

    table = soup.find('table', class_=table_class)
    if not table:
        print(f"Table with class '{table_class}' not found in the HTML file.")
        return

    with open(output_file, 'w', newline='', encoding='utf-8') as csvfile:
        csv_writer = csv.writer(csvfile)

        # Write header
        header = [th.text.strip() for th in table.find_all('th')]
        csv_writer.writerow(header)

        # Write rows
        for row in table.find_all('tr'):
            csv_writer.writerow([td.text.strip() for td in row.find_all('td')])

    print(f"Conversion completed. CSV file saved as '{output_file}'.")

def main():
    root = Tk()
    root.withdraw()

    # Ask for input HTML file
    input_file = filedialog.askopenfilename(filetypes=[('HTML Files', '*.html;*.htm')])
    if not input_file:
        print("No input file selected. Exiting.")
        return

    # Output file name based on input file
    output_file = input_file.rsplit('.', 1)[0] + '.csv'

    # Ask for table class
    table_class = input("Enter the class name of the table: ")

    # Convert HTML to CSV
    html_to_csv(input_file, output_file, table_class)

if __name__ == "__main__":
    main()
```

7.1.4 MergeTextFiles.py

```
# Title: MergeTextFiles.py
# Author: Rubinigg Michael
# Contributor: Rubinigg Michael
# Abstract: Merges text files into a new file based on user input
# Version 1.0
# Release date: 2023-12-28
# Place: Graz, Austria, EU
# Company: dr Mag. rer. nat. Michael Rubinigg
# Programming Language: Python 3.11
# Rights:  Creative Commons Attribution 4.0 (CC-BY)

import os
import tkinter as tk
from tkinter import filedialog

def merge_text_files(selected_files, output_directory, output_filename):
    merged_content = ""

    # Read content from selected text files
    for file_path in selected_files:
        with open(file_path, 'r') as file:
            merged_content += file.read() + '\n'

    # Create output directory if it doesn't exist
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)

    # Write merged content to the output text file
    output_path = os.path.join(output_directory, output_filename)
    with open(output_path, 'w') as output_file:
        output_file.write(merged_content)

    print(f"Merged files saved to: {output_path}")

def select_files():
    file_paths = filedialog.askopenfilenames(
        title="Select Text Files",
        filetypes=(("Text files", "*.*"), ("All files", "*.*"))
    )
    return file_paths

def select_output_directory():
    output_directory = filedialog.askdirectory(title="Select Output Directory")
    return output_directory

def get_output_filename():
    output_filename = input("Enter the output filename: ")
    return output_filename

def main():
    # Select text files to merge
    selected_files = select_files()

    if not selected_files:
        print("No files selected. Exiting.")
        return

    # Select output directory
    output_directory = select_output_directory()

    if not output_directory:
        print("No output directory selected. Exiting.")
        return

    # Get user-defined output filename
    output_filename = get_output_filename()

    # Merge selected text files and save to the output file
    merge_text_files(selected_files, output_directory, output_filename)

if __name__ == "__main__":
    main()
```