Batch Processing in Python

Overview

This document provides a guide to implementing **batch processing** for converting **AEM Forms (XFA/XML) to Forms.io JSON** using Python. The solution will work on **Windows, Mac, and Linux** and will support **parallel processing** for efficiency.

Steps to Ensure Cross-Platform Batch Processing

1. **Read all XDP files from a folder (`input_xdp/`)**
2. **Use multiprocessing for parallel processing**
3. **Convert XDP to JSON** (using a placeholder function)
4. **Store JSON output in `output_json/`**
5. **Log successes and errors into `logs/`**

Cross-Platform Compatibility

1. OS-Independent File Handling

- Different OS use different path formats:
  - **Windows**: Uses backslashes (`C:\Users\...`).
  - **Mac/Linux**: Uses forward slashes (`/home/user/...`).
- Use `os.path.join()` or `pathlib.Path()` instead of hardcoded paths.

2. Path Separators

- **Windows** uses `\` (backslash) in file paths.
- **Mac/Linux** use `/` (forward slash).
- Use `os.sep` to handle different path separators (`/` for Linux/Mac, `\` for Windows).

3. Script Execution Differences

- **Mac/Linux**: Use shebang (`#!/usr/bin/env python3`) at the start of the script.
- **Windows**: Ensure execution with `python` instead of `python3`.

4. Multiprocessing Compatibility

- Python's `multiprocessing` module has issues on Windows:
    - **Windows requires** `if __name__ == "__main__":` when using multiprocessing.
    - **Linux/Mac** don't have this restriction.

Batch Processing Code (with Error Handling & Logging)

Python

```python
import os
import json
import logging
import multiprocessing
from pathlib import Path
from lxml import etree
import report  # Import report module for success/error logging

# Setup paths (Cross-Platform)
BASE_DIR = Path(__file__).parent
INPUT_DIR = BASE_DIR / "input_xdp"
OUTPUT_DIR = BASE_DIR / "output_json"
LOG_DIR = BASE_DIR / "logs"

# Ensure output directories exist
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)
LOG_DIR.mkdir(parents=True, exist_ok=True)

# Setup logging
logging.basicConfig(
    filename=LOG_DIR / "batch_processing.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)

# Function to convert XDP to JSON
def convert_xdp_to_json(xdp_file):
    try:
        xdp_file = Path(xdp_file)  # Ensure it's a Path object
```

```
        # Read XDP (XML format)
        with open(xdp_file, "r", encoding="utf-8") as file:
            xml_data = file.read()

        # Parse XML (Handling errors safely)
        try:
            root = etree.fromstring(xml_data)
        except etree.XMLSyntaxError as e:
            logging.error(f"XML Parsing Error in {xdp_file}:
{e}")
            report.report_error(xdp_file, f"XML Parsing Error:
{e}")
            return

        # Extract metadata safely
        form_name = root.attrib.get("name", "Unknown")
        last_modified = root.attrib.get("lastModified",
"Unknown")

        json_data = {
            "form_name": form_name,
            "lastModified": last_modified,
            "fields": []
        }

        # Generate output filename
        output_file = OUTPUT_DIR / f"{xdp_file.stem}.json"

        # Write JSON output
        with open(output_file, "w", encoding="utf-8") as
json_file:
            json.dump(json_data, json_file, indent=4)

        # Get file metadata
        timestamp = os.path.getmtime(xdp_file)  # File's last
modified timestamp
```

```
        # Log success using report.py
        report.report_success(timestamp, last_modified,
json_data)

        logging.info(f"Successfully converted: {xdp_file}")

    except Exception as e:
        logging.error(f"Error processing {xdp_file}: {e}")
        report.report_error(xdp_file, str(e))

# Batch processing function
def process_files():
    xdp_files = list(INPUT_DIR.glob("*.xdp"))  # Get all XDP
files

    if not xdp_files:
        logging.warning("No XDP files found in the input
directory.")
        print("No XDP files found.")
        return

    print(f"Processing {len(xdp_files)} files...")

    # Use multiprocessing for parallel execution
    with
multiprocessing.Pool(processes=multiprocessing.cpu_count()) as
pool:
        pool.map(convert_xdp_to_json, xdp_files)

    print("Batch processing completed.")

# Ensure compatibility with Windows multiprocessing
if __name__ == "__main__":
    process_files()
```

How Error Handling & Logging Works

1. **Try-Except Blocks**:
    - If an error occurs in `convert_xdp_to_json()`, it is caught and logged instead of stopping the script.
    - Example: If a file is corrupted, it will be logged, and processing will continue for other files.
2. **Logging Levels**:
    - **INFO**: Logs successful conversions.
    - **WARNING**: Logs when no XDP files are found.
    - **ERROR**: Logs errors encountered during conversion.
3. **Log File Location**:
    - All logs are stored in the `logs/` directory under `batch_processing.log`.

How to Run the Script?

1. Place all `.xdp` files inside the `input_xdp/` folder.
2. Run the script on any OS:
    - **Windows**: `python batch_converter.py`
    - **Mac/Linux**: `python3 batch_converter.py`
3. Converted JSON files will be stored in the `output_json/` folder.
4. Logs are saved in the `logs/` folder.