Reports and Error Handling Documentation

1. Introduction

This document provides an overview of the reporting and error-handling mechanisms in the XML to JSON conversion tool. It covers how errors are logged, how reports are generated, and the structure of the reporting system.

2. Report Generation

2.1 Purpose

The reporting system logs successful conversions, errors, and cases requiring manual intervention. This helps in debugging, tracking conversion accuracy, and identifying patterns in failures.

2.2 Report Structure

Each conversion run generates a JSON report with the following structure:

```
Unset
{
    "success": [],
    "errors": [],
    "manual_intervention_needed": []
}
```

2.3 Report Filename Convention

Reports are generated with a timestamped filename format:

```
Unset
<xml_filename>_report_<YYYYMMDD_HHMMSS>.json
```

This ensures that every run creates a new report rather than overwriting previous logs.

2.4 Logging Success Entries

When a field is successfully converted, the following entry is added to the success section:

```
Unset
{
    "field_in_original_form": "XMLFieldName",
    "converted_field": "JSONFieldName",
    "value": "ConvertedValue",
    "status": "Converted successfully"
}
```

3. Error Handling

3.1 Types of Errors Logged

- **Parsing Errors**: Issues while reading the XML.
- **Field Mapping Errors**: Missing or incorrect mappings.
- **Data Extraction Errors**: Missing expected values in the XML.
- **Validation Errors**: Incorrectly formatted or incomplete data.

3.2 Error Log Structure

Errors are stored in the errors section of the report with details:

```
Unset
{
    "field_in_original_form": "XMLFieldName",
    "converted_field": "JSONFieldName",
    "value": "ExtractedValueOrNull",
    "issue": "Description of the error",
    "status": "Conversion failed"
}
```

3.3 Manual Intervention Logging

If a field cannot be automatically mapped and requires review, it is logged under manual_intervention_needed:

```
Unset
{
    "field_in_original_form": "XMLFieldName",
    "converted_field": "JSONFieldName",
    "value": "ExtractedValueOrNull",
    "status": "Needs manual review"
}
```

4. Integration into Conversion Process

4.1 Where Logging is Called

- **Success Logging**: When a field is successfully converted.
- **Error Logging**: When an issue is encountered during parsing or mapping.
- **Manual Review Logging**: When a field cannot be automatically converted.

4.2 Implementation Example

```
Python
report.report_success("XMLFieldName", "JSONFieldName",
extracted_value)
report.report_error("XMLFieldName", "JSONFieldName",
extracted_value, "Missing mapping")
report.report_manual_intervention("XMLFieldName",
"JSONFieldName", extracted_value)
```

5. Batch Processing Considerations

5.1 Differences in Batch vs. Single-File Processing

Feature	Single-File Processing	Batch Processing
Execution Flow	Stops or continues based on error type.	Continues processing remaining files.

Error Logging	Logs errors for that single file.	Logs errors separately for each file.
Impact of Errors	Can prevent successful conversion.	Affects only specific files, others proceed.
Report Generation	One report per execution.	One report per file (or a consolidated report).

5.2 How Batch Processing Handles Errors

- Each **XML file** is processed **independently**.
- If a file fails to convert:
  - The error is logged **without stopping execution**.
  - Other files continue processing.
- Reports are generated **per file**, ensuring traceability.
- **Severe errors** (e.g., invalid XML format) are flagged but don't halt the batch.

5.3 Batch Report Structure

For batch processing, we can generate either **individual reports per file** or a **consolidated report** summarizing all files:

Per File Reports:

Unset
```
<xml_filename>_report_<YYYYMMDD_HHMMSS>.json
```

Consolidated Batch Report:

Unset
```
{
    "processed_files": [
        {
            "file": "file1.xml",
            "status": "success",
            "errors": 2
        },
        {
            "file": "file2.xml",
            "status": "failed",
            "errors": 5
        }
    ],
    "total_files": 10,
    "successful_conversions": 8,
    "failed_conversions": 2
}
```

5.4 Summary

✅ Batch processing ensures that **one file's failure doesn't stop the others**. ✅ **Errors are logged per file** to maintain traceability. ✅ A consolidated **summary report** can be generated if needed.

Would you like a **consolidated batch summary report feature**, or is per-file reporting enough? 🚀