**team emmy SDK**

## **Overview**

The **EIDR Language Processing SDK** is a Python package that enables efficient processing of EIDR (Entertainment IDentifier Registry) records. It provides tools for:

* Retrieving metadata from EIDR
* Performing language identification, translation, and transliteration
* Processing titles using AWS Lambda or OpenAI
* Batch processing and saving results

## **How this Data Flow?**

The chosen data flow supports robustness, scalability, and integration with modern cloud platforms. Here's how and why the flow works:

### Purpose

EIDR records can have titles in various languages, including non-Latin scripts. To make these usable in international systems, the SDK provides a standard way to:

* Identify the language of a title.
* Translate non-English titles to English.
* Transliterate non-Latin scripts to Latin characters.
* Score language matches for QA or tagging.

### Step-by-Step System Workflow:

1. **Input**: List of EIDR IDs or an individual EIDR ID.
2. **Metadata Retrieval**: Uses *EIDRClient* to fetch XML metadata from the EIDR registry via RESTful API.
3. **Title Extraction**: Extracts all available titles (primary and alternate) with associated language codes.
4. **Language Detection**: The system detects the language using AWS Lambda or OpenAI backend.
5. **Translation and Transliteration**: Applies necessary transformations if the title is not in English or not in Latin script.
6. **Scoring (optional)**: Evaluates how well a title fits an expected language.
7. **Saving Results**: Outputs result into JSON files for easy parsing, reporting, or insertion into downstream databases.

### Benefits of the Flow:

* **Separation of Concerns**: Each step (fetch, process, save) is independently testable and configurable.
* **Asynchronous/Concurrent Capabilities**: BatchProcessor allows for multi-threaded execution.
* **Cloud-Ready**: Designed to integrate with AWS services for secure, scalable processing.
* **Fallback Support**: If AWS Lambda fails, OpenAI APIs act as a secondary processing engine.

## **Query Design**

EIDR records are fetched via HTTP GET requests to the EIDR registry’s XML-based API. Sample query structure:

Examples request:

*<?xml version="1.0" encoding="UTF-8"?>*

*<Request xmlns="http://www.eidr.org/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">*

*<Operation>*

*<Query>*

*<Expression>({query})</Expression>*

*<PageNumber>{QueryPageOffset}</PageNumber>*

*<PageSize>{requestPagesize}</PageSize>*

*</Query>*

*</Operation>*

*</Request>*

Titles are extracted using XPath expressions:

*root.findall('.//{http://www.eidr.org/schema}AlternateTitle')*

These are then filtered by attributes like xml:lang and titleType.

By separating the data retrieval from the processing steps, the SDK allows users to work with either live EIDR data or pre-saved XML files for debugging or offline work.

All queries are designed customized to extracted the requested data. For example, the following query is designed to get untitled titles:

(((/FullMetadata/BaseObjectData/ResourceName "und") OR (/FullMetadata/BaseObjectData/AlternateResourceName "und")) AND /FullMetadata/BaseObjectData/Status "valid")

## **Components**

**1. Config**

Handles configuration using environment variables or .env files.

* Loads credentials and settings for EIDR, OpenAI, and AWS Lambda
* Validates required configurations
* Logs available processing methods

**2. EIDRClient**

Fetches and parses metadata from the EIDR registry.

* get\_metadata(eidr\_id) – Fetch XML metadata from EIDR
* extract\_titles(xml\_string) – Extracts primary and alternate titles with language data

**3. LanguageProcessor**

Processes language-related tasks via OpenAI or AWS Lambda.

* identify\_language(title) – Detects language and confidence score
* translate\_only(title, source\_language) – Translates to English
* transliterate\_only(title, source\_language) – Converts to Latin script
* score\_language(title, expected\_language) – Validates if text matches a target language
* process\_title(record\_id, title, original\_language) – Full processing pipeline

**4. FileProcessor**

Handles file operations related to ID input and output.

* get\_ids\_from\_file(path) – Reads IDs from a file
* save\_ids(ids, folder) – Saves list of IDs to a timestamped file
* save\_results\_to\_file(results, path) – Saves JSON results

**5. BatchProcessor**

Executes multi-ID processing using thread pools.

* process\_single\_id(eidr\_id) – Processes a single EIDR ID end-to-end
* process\_batch(ids, output\_file) – Processes a batch of EIDR IDs concurrently

**6. EIDRSDK**

Main interface for users integrating the SDK into their tools or scripts.

* identify\_language(title)
* translate(title, source\_language)
* transliterate(title, source\_language)
* score\_language(title, expected\_language)
* process\_single\_id(eidr\_id)
* process\_id\_file(file\_path, output\_file, confirm)

## **Environment Configuration**

A .env file or set environment variables directly:

*EIDR\_REGISTRY\_KEY=your\_registry\_key*

*EIDR\_LOGIN=your\_login*

*EIDR\_PARTYID=your\_party\_id*

*EIDR\_PASSWORD=your\_password*

*OPENAI\_API\_KEY=your\_openai\_key*

*AWS\_REGION=us-east-1*

*LAMBDA\_FUNCTION\_NAME=your\_lambda\_function*

*REQUEST\_TIMEOUT=10*

*MAX\_WORKERS=10*

## **Logging**

Logs are saved to eidr\_language.log with timestamps and levels (INFO, WARNING, ERROR).

## **Notes**

* AWS Lambda is preferred for larger workloads or secure deployments.
* Transliteration and scoring are language-agnostic and work with mixed-language content.