AWS Lambda Language Processing Service

# Overview

This AWS Lambda function is a serverless backend component of the TEAM EMMY SDK project. Its primary purpose is to process titles from the Entertainment IDentifier Registry (EIDR) by:

* Detecting their language
* Transliterating them to Latin script if necessary
* Translating them to English

The function is optimized for efficient execution and designed to **minimize repeated API calls** by **caching processed results** in an Amazon DynamoDB table.

# Functional Objectives

* **Language Detection**: Identifies the original language using OpenAI's large language models and returns both the language name and ISO 639 code.
* **Transliteration**: Converts the input title to Latin-1 characters for compatibility with Latin-based systems.
* **Translation**: Produces a fluent English version of the title.
* **Confidence Scoring**: Returns a confidence score (0 to 1) for the language detection.
* **Caching with DynamoDB**: Avoids repeated API calls by checking and storing processed results.

# System Workflow

**1. Input Event**  
The function expects the following fields:

* **RecordID** (string): Unique EIDR identifier
* **Title** (string): Content title
* **OriginalLanguage** (string, optional): Original metadata language

**2. Check for Existing Record**  
If the record already exists in DynamoDB (based on RecordID and Title), the cached result is returned.

**3. Language Processing via OpenAI**  
If no record is found, the function sends a structured prompt to OpenAI’s GPT-4o-mini for:

* Language detection
* Transliteration
* Translation
* Confidence scoring

**4. Store Results in DynamoDB**  
Parsed results are saved for future reference using the put\_item method.

**5. Return Processed Output**  
A JSON response is returned, with a message indicating if the result was cached or newly processed.

# Core Components

1. **lambda\_handler(event, context)**
   * Entry point for Lambda
   * Controls flow and calls sub-functions
2. **get\_record(dynamodb\_client, record\_id, title)**
   * Looks up existing record in DynamoDB
3. **process\_with\_openai(title)**
   * Calls OpenAI API to process the title
   * Returns structured result with language, code, confidence, transliteration, translation
4. **store\_record(dynamodb\_client, record\_id, title, original\_language, data)**
   * Saves the processed record
   * Converts numeric confidence to string (DynamoDB compatibility)

# Environment Variables

* openai\_key: OpenAI API Key
* AWS credentials are injected automatically via IAM role (no hardcoding needed)

# DynamoDB Table Schema

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| RecordID | String | Partition key |
| Title | String | Sort key |
| OriginalLanguage | String | Original language (if provided) |
| DetectedLanguage | String | Language name detected by OpenAI |
| DetectedCode | String | ISO 639-1 or 639-3 code |
| Confidence | String | Confidence score (0–1) as string |
| Transliteration | String | Latin-script version of the title |
| Translation | String | English translation of the original title |

# Error Handling

* Missing RecordID or Title: Returns **400 Bad Request**
* OpenAI/DynamoDB failure: Returns **501 Internal Error** with details

**Example Request Event**

{

"RecordID": "10.5240/1234-5678-ABCD-EFGH-9876-C",

"Title": "天空の城ラピュタ",

"OriginalLanguage": "ja"

}

**Example Response**

{

"message": "Record processed and stored.",

"data": {

"DetectedLanguage": "Japanese",

"ISO639LanguageCode": "ja",

"Confidence": "0.97",

"Transliteration": "Tenkū no Shiro Rapyuta",

"Translation": "Castle in the Sky"

}

}