**Title Language Processor Lambda Documentation**

**Overview**

The Title Language Processing Lambda is an AWS Lambda-based Python service designed to process movie titles. It leverages OpenAI’s API for language detection using the GPT 4.0 mini, for the process of the transliteration, and translation as well. The service also integrates with AWS DynamoDB to store and retrieve processed records efficiently.

**Purpose**

Movie titles, especially from international registries such as the EIDR registry, may appear in multiple languages and scripts. This function standardizes such data through:

* **Language Detection** (e.g., detect any language from what it is, for example: French from “Bonjour”)
* **Transliteration** (convert to Latin characters)
* **Translation** (e.g., translate to English)
* **Confidence Scoring** (rate certainty of language detection)
* **Caching** of processed results in **DynamoDB** to avoid redundant processing.

**System Workflow**

1. **Input**

A JSON payload with the following fields:

* RecordID – Unique identifier for the content
* Title – The movie title (string)
* OriginalLanguage – Optional field representing the expected language

1. **Record Lookup**

* Checks DynamoDB for an existing record using RecordID and Title.
* If found, returns the stored data immediately.

1. **Language Processing**

* Uses the OpenAI API to detect language, provide a transliteration, and translate the title.
* GPT returns a structured JSON output with language name, detected language, detected code, confidence score, transliteration, and English translation.

1. **Store & Respond**

* Save the results to DynamoDB with composite key (RecordID, Title).
* Returns the response as JSON

**Key Components**

**Lambda Entry:**

**def lambda\_handler(event, context):**

* Handles event parsing, validation, and orchestrates all sub-processes.
* Handles exceptions gracefully and logs detailed errors.

**def get\_record(dynamodb\_client, record\_id, title):**

* Checks if the record already exists in DynamoDB.
* Uses a composite key of RecordID and Title.

**def process\_with\_openai(title):**

Sends a structured prompt to OpenAI using gpt-4o-mini to:

* Detect language and confidence
* Transliterate into Latin
* Translate into English
* The prompt ensures output is always JSON-formatted.

**Through something like this with the process with OpenAI, that is displayed in the code:**  
 *# System and User Prompts as messages*

*system\_prompt = "You are an expert language model trained to detect languages, provide transliterations, and generate translations for movie titles. Provide a confidence score between 0 and 1 for the language detection."*

*user\_prompt = f"Given the title: '{title}', detect the language both by name and ISO 639 language code, provide a transliteration into the Latin-1 character set, and translate it to English. Return the output in JSON format with keys: DetectedLanguage, ISO639LanguageCode, Confidence, Transliteration, and Translation. Do not include code block formatting."*

*# Making the API call using the updated method*

*response = openai.chat.completions.create(*

*model="gpt-4o-mini",*

*messages=[*

*{"role": "system", "content": system\_prompt},*

*{"role": "user", "content": user\_prompt}*

*],*

*temperature=0 # Deterministic output*

*)*

**def store\_record(dynamodb\_client, record\_id, title, original\_language, data):**

Stores OpenAI-processed data in DynamoDB using:

* RecordID (PK) – Partition Key
* Title (SK) – Sort Key
* Attributes: OriginalLangauge, DetectedLanguage, DetectedCode, Confidence (as string), Transliteration, Translation

**Sample Output:***{*

*"DetectedLanguage": "French",*

*"DetectedCode": "fr",*

*"Confidence": 0.98,*

*"Transliteration": "Bonjour",*

*"Translation": "Hello"*

*}*

**Environment Configuration**

Values are read from environment variables set within Lambda, or can also be set within the IDE that you are using:

| **Variable Name** | **Description** |
| --- | --- |
| openai\_key | OpenAI API key |
| AWS\_REGION | AWS region of DynamoDB |
| TitleLanguageData | DynamoDB table name (hardcoded) |
|  |  |

**Error Handling**

* **400 Bad Request:** If input lacks RecordID or Title.
* **501 Internal Error:** Captures all unexpected exceptions.
* **DynamoDB Error:** Captures AWS client-side exceptions.
* **OpenAI Error:** If GPT response is abnormally formed or the API fails.

**Logging**

* Logging to AWS via Lambda standard output.
* Errors and responses are available for audit or monitoring.

**Notes**

* OpenAI output must strictly follow JSON formatting. GPT model version (gpt-4o-mini) is chosen for performance and quality.
* Transliteration and translation are language-agnostic.
* Future enhancements could include improving processing speed and tracking processing latency.
  + Focusing on the language tool, to shoot out more accurate results.