SDK for Metadata Bank API v1 - "getApiTokenEncrypted"- AWS Lambda Node JS Implementation

The Metadata Bank API v1 introduces breaking changes, including the implementation of OAuth 2 for API key access token management. To assist developers in transitioning to this new version, we provide an SDK that simplifies the process of obtaining and managing access tokens. This document provides a detailed explanation of the SDK's functionalities.

Overview

The SDK includes a main function, `refreshToken`, which is responsible for obtaining an access token from the Metadata Bank API. This function also manages the token's lifecycle, ensuring that the token is valid and refreshing it when necessary. The SDK is designed to work within an AWS Lambda function and uses *AWS Secrets Manager* to securely store the consumer key and consumer secret. It also uses *AWS DynamoDB* to store the access token.

refreshToken Function

The `refreshToken` function is a critical part of the SDK. It is responsible for retrieving an access token, which is required for making authenticated requests to the API. The function also keeps track of the expiration time of the token.

Functionality

The `refreshToken` function performs the following steps:

1. It first retrieves the consumer key and consumer secret from AWS Secrets Manager.
2. It then checks if a valid token is already available in the DynamoDB table. The token is considered valid if it has not yet expired. The function reads the token from the table and decrypts it using a key generated from the consumer key and consumer secret.
3. If a valid token is not available in the table (either the table does not exist, or the token has expired), the function will attempt to fetch a new token from the API.

4. The function makes a POST request to the API's token endpoint, passing the consumer key and consumer secret as part of the request headers.
5. If the request is successful, the API will return a new access token along with its expiration time. The function calculates the expiry timestamp (in milliseconds) and adds it to the token object.
6. The function then saves the token object (including the expiry timestamp) to the DynamoDB table. The token object is encrypted using the same key as before, and then written to the table.
7. If the request fails, the function will retry the request a certain number of times (defined by the `retries` parameter) with a delay between each attempt (defined by the `interval` parameter). If all retries fail, the function will throw an error.
8. Finally, the function returns the token object (including the expiry timestamp).

This process ensures that the SDK always has a valid access token to use for API requests. By saving the token to a DynamoDB table, the SDK can minimize the number of requests made to the API's token endpoint.

Parameters

The `refreshToken` function accepts the following parameters:

- `retries` (optional): The number of times to retry the request if it fails. Defaults to 2.
- `interval` (optional): The delay (in milliseconds) between each retry attempt. Defaults to 3000.
- `shouldRetry` (optional): A boolean indicating whether the function should retry the request if it fails. Defaults to true.

Environment Variables

The `refreshToken` function uses the following environment variables:

- `RETRIES`: The default number of retries.
- `INTERVAL`: The default retry interval.
- `SECRET_NAME`: The name of the secret in AWS Secrets Manager that contains the consumer key and consumer secret.
- `URL`: The URL of the API's token endpoint.
- `SCOPE`: The scope of the access token.

Return Value

The `refreshToken` function returns a Promise that resolves to an object containing the access token, its expiration time (in seconds), and its expiry timestamp (in milliseconds). If the function fails to retrieve a token after all retries, it will reject the Promise with an error.

Additional Functions

The SDK also includes several helper functions:

- `getSecrets`: Retrieves secrets from AWS Secrets Manager.
- `generateEncryptionKey`: Generates an encryption key from the consumer key and consumer secret.
- `encrypt`: Encrypts data using a given key.
- `decrypt`: Decrypts data using a given key.
- `saveTokenToDynamoDB`: Saves an encrypted token to a DynamoDB table.
- `loadTokenFromDynamoDB`: Loads an encrypted token from a DynamoDB table and decrypts it.
- `fetchToken`: Makes a POST request to the API's token endpoint to fetch a new token.
- `retryFetchToken`: Retries the `fetchToken` function a specified number of times with a delay between each attempt.

These functions are exported from the SDK and can be used independently if needed. However, in most cases, developers will only need to use the `refreshToken` function.

Usage

To use the SDK, import the `refreshToken` function and call it when you need an access token:

```
const { refreshToken } = require('./getApiTokenEncrypted');

exports.handler = async (event, context) => {
    try {
        const token = await refreshToken();
        // Use the token to make an API request...
    } catch (error) {
        console.error('Failed to refresh token:', error);
        // Handle the error...
    }
};
```

Remember to set the necessary environment variables before running your application.

Additional Notes

Ensure that your Lambda function has the necessary permissions to access AWS Secrets Manager and DynamoDB. This includes:

- `secretsmanager:GetSecretValue` permission for the secret that contains your consumer key and consumer secret.

- `dynamodb:PutItem`, `dynamodb:GetItem`, and `dynamodb:UpdateItem` permissions for the DynamoDB table that stores your access tokens.

You can set these permissions in the IAM role that is associated with your Lambda function.

Also, remember to set the `SECRET_NAME` environment variable to the name of your secret in AWS Secrets Manager. The SDK uses this variable to retrieve your consumer key and consumer secret.

Conclusion

The `getApiTokenEncrypted` SDK simplifies the process of managing access tokens for the Metadata Bank API. By handling the complexities of token retrieval, encryption, storage, and lifecycle management, the SDK allows developers to focus on making API requests and building their applications.

Prepared by: Patrick Roch

Public Broadcasting Service(PBS)