Amazon Translate

개발자 안내서

Amazon Translate: 개발자 안내서

Table of Contents

Amazon Translate란 무엇입니까?

Amazon Translate는 텍스트 번역 서비스로 고급 머신 러닝 기술을 사용하여 우수한 품질의 번역을 온디맨드 방식으로 제공합니다. 이 서비스를 사용하여 구조화되지 않은 텍스트 문서를 번역하거나, 여러 언어로 작업하는 애플리케이션을 만들 수 있습니다.

주제

지원되는 언어

Amazon Translate는 다음 표에 나열된 모든 언어로 문서를 번역 할 수 있습니다.

동아시아어	게르만아	로망스어	슬라브어	셈어	인도이란어	피노우그리아어	튀르크어	
• 중국어(간체) • 중국어 번체 • 인도네시아어 • 일본어 • 한국어 • 말레이어	• 덴마크어 • 네덜란드어 • 독일어 • 영어 • 노르웨이어 • 스웨덴어	• 프랑스어 • 이탈리아어 • 스페인어 • 포르투갈어	• 체코어 • 폴란드어 • 러시아어	• 아랍어 • 히브리어	• 힌디어 • 페르시아어	• 핀란드어	• 터키어	

Note

힌디어, 말레이어, 노르웨이어 및 페르시아어는 AWS GovCloud(미국 서부), 아시아 태평양(뭄바이), 아시아 태평양(싱가포르), 아시아 태평양(도쿄), 캐나다(중부) 리전에서 사용할 수 없습니다.

언어 쌍

Amazon Translate 소스 언어(입력 언어)와 대상 언어(출력 언어) 간 번역을 제공합니다. 소스 언어-대상 언어 조합을 언어 쌍이라고 합니다.

모든 언어 쌍은 다음을 제외하고 Amazon Translate에서 지원됩니다.

지원되지 않는 언어 쌍

- 중국어(간체)-중국어(번체)
- 중국어(번체)-중국어(간체)
- 한국어-히브리어
- 노르웨이어-아랍어

- 노르웨이어-히브리어

Amazon Translate 사용 사례

Amazon Translate가 비즈니스 필요 사항을 충족할 수 있는 다양한 시나리오가 있습니다. 예를 들어 다음 작업을 할 수 있습니다.

- Amazon Translate을 애플리케이션에 통합하여 다국어 사용자 환경 지원
  - 회의록, 기술자 보고서, 기술 자료 문서, 게시물 등 회사에서 작성한 콘텐츠를 번역합니다.
  - 이메일, 게임 중 채팅, 고객 서비스 채팅 등 사람들 간의 의사 소통을 번역하여 고객과 직원이 원하는 언어로 접속할 수 있도록 합니다.
- Amazon Translate을 수신 데이터에 대한 회사 워크플로우의 일부로 사용
  - 소셜 미디어와 뉴스 피드 등 다양한 언어로 된 텍스트를 분석합니다.
  - eDiscovery 사례 등 다양한 언어로 된 정보를 검색합니다.
- Amazon Translate을 다른 AWS 서비스와 통합하여 언어에 구애받지 않는 프로세스 실행
  - Amazon Comprehend와 함께 사용하여 소셜 미디어 스트림과 같은 구조화되지 않은 텍스트에서 명명된 엔터티, 감정 및 핵심 구절을 추출해 냅니다.
  - Amazon Transcribe와 함께 사용하여 자막 및 라이브 자막을 여러 언어로 제공합니다.
  - Amazon Polly와 함께 사용하여 번역된 콘텐츠 읽어주기
  - Amazon S3와 함께 사용하여 문서 리포지토리를 번역합니다.
  - Amazon DynamoDB, Amazon Aurora 및 Amazon Redshift와 함께 사용하여 데이터베이스에 저장된 텍스트를 번역합니다.
  - AWS Lambda 또는 AWS Glue와 함께 사용하여 워크플로우를 원활하게 통합합니다.

이것은 HIPAA 적격 서비스입니다. AWS, 미국 HIPAA(Health Insurance Portability and Accountability Act of 1996), AWS 서비스를 이용한 보호 대상 건강 정보(PHI)의 처리, 저장, 전송에 대한 자세한 정보는 HIPAA 개요를 확인하십시오.

Amazon Translate를 처음 사용하십니까?

처음 사용하는 경우, 먼저 다음 단원을 순서대로 읽어보십시오.

1. Amazon Translate 작동 방식 (p. 3)—는 Amazon Translate을 소개합니다.
2. Amazon Translate 시작하기 (p. 5)—AWS 계정을 설정하고 Amazon Translate를 테스트하는 방법을 설명합니다.
3. 예제 (p. 20)—Java와 Python 코드 예제를 제공합니다. 이를 사용하여 Amazon Translate의 작동 방식을 살펴볼 수 있습니다.
4. API Reference (p. 59)— Amazon Translate 작업에 대한 참조 설명서를 소개합니다.

Amazon Translate 작동 방식

Amazon Translate 서비스는 언어 번역을 위해 교육된 신경망을 기반으로 합니다. 이러한 기반에서 소스 언어(번역되는 텍스트의 원래 언어)와 대상 언어(텍스트가 번역되는 결과 언어) 간에 번역할 수 있습니다. 영어를 소스 언언 대상 언어로 사용할 필요는 없지만 Amazon Translate에서 모든 언어 조합이 다 지원되지는 것은 아닙니다. 자세한 내용은 언어 쌍 (p. 1) 단원을 참조하십시오.

Amazon Translate로 작업할 때는 소스 텍스트를 제공하고 출력 텍스트를 얻게 됩니다.

- 소스 텍스트— 번역하려는 텍스트입니다. 소스 텍스트는 UTF-8 형식으로 입력합니다.
- 출력 텍스트— Amazon Translate가 대상 언어로 번역한 출력 텍스트. 출력 텍스트도 UTF-8 형식입니다. 소스 언어와 대상 언어에 따라 출력 텍스트의 문자 수가 입력 텍스트보다 많을 수 있습니다.

이 번역 모델에는 인코더와 디코더라는 두 가지 구성 요소가 있습니다. 인코더는 소스 문장을 한 번에 한 단어씩 읽고, 소스 텍스트의 뜻을 담은 의미 표현을 구성해 냅니다. 디코더는 의미 표현을 사용하여 한 번에 한 단어씩 대상 언어로 번역문을 만듭니다.

Amazon Translate는 주의 메커니즘을 통해 문맥을 이해합니다. 이 메커니즘은 소스 텍스트에서 가장 중요한 단어를 판단하여 이어지는 대상 단어를 찾아 줍니다. 디코더는 주의 메커니즘을 토대로 소스 문장에서 가장 중요한 부분에 집중하게 됩니다. 따라서 디코더는 모호한 단어나 구절을 정확하게 번역할 수 있습니다.

이 모델에서 생성된 대상 단어가 디코더에 입력됩니다. 그리고 네트워크는 문장의 끝에 도달할 때까지 계속해서 단어를 만들어 냅니다.

텍스트를 번역하려면 TranslateText (p. 71) 메서드를 호출하고, 아래 표에 나열된 언어 코드로 소스 언어와 대상 언어를 지정하면 됩니다.

언어	코드
아랍어	ar
중국어 간체	zh
중국어 번체	zh-TW
체코어	cs
덴마크어	da
네덜란드어	nl
영어	en
핀란드어	fi
프랑스어	fr
독일어	de
히브리어	he
힌디어	hi
인도네시아어	id
이탈리아어	it

언어	코드
일본어	ja
한국어	ko
말레이어	ms
노르웨이어	no
페르시아어	fa
폴란드어	pl
포르투갈어	pt
러시아어	ru
스페인어	es
스웨덴어	sv
터키어	tr

언어 자동 감지

Amazon Translate는 소스 텍스트의 언어를 자동으로 감지할 수 있습니다. 자동 언어 감지를 위해 소스 텍스트를 제공할 때 `auto`를 소스 언어로 지정합니다. Amazon Translate가 사용자를 대신해 Amazon Comprehend를 호출하여 소스 언어에 사용된 언어를 결정합니다. 사용자는 언어 자동 감지를 선택함으로써 Amazon Comprehend 서비스 약관 및 계약에 동의하게 됩니다. Amazon Comprehend의 요금에 대한 자세한 내용은 Amazon Comprehend요금 을 참조하십시오.

예외 처리

지원되지 않는 소스 언어나 대상 언어를 지정하면 Amazon Translate가 다음 예외를 반환합니다.

- UnsupportedLanguagePairException – Amazon Translate는 지원되는 언어 간에 번역을 지원합니다. 언어 쌍 간의 번역이 지원되지 않으면 예외가 반환됩니다. 자세한 내용은 언어 쌍 (p. 1) 단원을 참조하십시오.
- DetectedLanguageLowConfidenceException – 언어 자동 감지를 사용할 때 Amazon Translate가 올바른 소스 언어를 감지했는지 확신할 수 없으면 이 예외가 반환됩니다. 신뢰도가 낮아도 무방한 경우에는 예외에 반환된 소스 언어를 사용하면 됩니다.

다음 단계

지금까지 Amazon Translate의 작동 방식에 대해 배웠습니다. 다음 단원에서는 솔루션을 생성하는 방법에 대해 알아보겠습니다.

- Amazon Translate 시작하기 (p. 5)
- 예제 (p. 20)

Amazon Translate 시작하기

Amazon Translate 사용을 시작하려면 AWS 계정을 설정하고 AWS Identity and Access Management(IAM) 사용자를 만듭니다. AWS Command Line Interface(AWS CLI)를 사용하려면 다운로드하여 구성합니다.

주제

1단계: AWS 계정 설정 및 관리자 사용자 만들기

Amazon Translate를 처음 사용하는 경우, 먼저 다음 작업을 완료해야 합니다.

AWS에 가입

Amazon Web Services(AWS) 가입 시 AWS 계정은 Amazon Translate를 포함한 모든 AWS 서비스에 자동으로 등록되며, 사용한 서비스에 대해서만 청구됩니다.

Amazon Translate에서는 사용한 리소스에 대해서만 비용을 지불합니다. AWS를 처음 사용하는 고객인 경우 Amazon Translate를 무료로 시작할 수 있습니다. 자세한 정보는 AWS 프리 티어를 참조하십시오.

AWS 계정을 이미 가지고 있다면 디음 단원으로 건너뛰십시오.

AWS 계정을 생성하려면

1. https://aws.amazon.com/을 열고 Create an AWS Account(AWS 계정 생성)를 선택합니다.

   Note

   전에 AWS 계정 루트 사용자 자격 증명을 사용하여 AWS Management 콘솔 에 로그인한 적이 있는 경우 Sign in to a different account(다른 계정으로 로그인)를 선택합니다. 전에 IAM 자격 증명을 사용하여 콘솔에 로그인한 적이 있는 경우 Sign-in using root account credentials(루트 계정 자격 증명으로 로그인)를 선택합니다. 그런 다음 Create a new AWS account(새 AWS 계정 생성)를 선택합니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화를 받고 전화 키패드를 사용하여 확인 코드를 입력하는 과정이 있습니다.

다음 작업에 필요하므로 AWS 계정 ID를 메모해 두십시오.

IAM 사용자 생성

Amazon Translate 같은 AWS 서비스를 사용하려면 액세스할 때 자격 증명을 제공해야 합니다. 서비스는 이를 통해 사용자가 서비스 리소스에 액세스할 수 있는 권한이 있는지 확인할 수 있습니다.

AWS 계정의 자격 증명을 사용하지 말고 AWS Identity and Access Management(IAM)을 사용하여 AWS에 액세스하십시오. IAM을 사용하여 AWS에 액세스하려면 IAM 사용자를 만들고 이 사용자를 관리자 권한이 있는 IAM 그룹에 추가한 후 해당 IAM 사용자에게 관리자 권한을 부여하십시오. 그러면 특정 URL이나 IAM 사용자의 자격 증명을 사용하여 AWS에 액세스할 수 있습니다.

이 가이드의 연습 예제는 `adminuser`라고 하는 관리자 권한을 가진 IAM 사용자가 있다는 전제하에서 진행됩니다.

관리자 를 만들려면

- AWS 계정에서 `adminuser`라는 관리자를 만듭니다. 지침은 IAM 사용 설명서의 첫 번째 IAM 사용자 및 관리자 그룹 생성 단원을 참조하십시오.

IAM에 대한 자세한 내용은 다음을 참조하십시오.

- AWS Identity and Access Management(IAM)
- 시작하기
- IAM 사용 설명서

다음 단계

2단계: AWS Command Line Interface(AWS CLI) 설정 (p. 6)

2단계: AWS Command Line Interface(AWS CLI) 설정

AWS CLI를 사용하여 Amazon Translate에 대한 대화식 호출을 생성합니다.

AWS CLI를 설정하려면

1. AWS CLI을(를) 다운로드하고 구성합니다. 지침은 AWS Command Line Interface 사용 설명서에서 다음 항목을 참조하십시오.
   - AWS Command Line Interface를 이용한 설정
   - AWS Command Line Interface 구성
2. AWS CLI `config` 파일에서 관리자를 위한 지정된 프로필을 추가합니다.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

   이 프로필은 AWS CLI 명령을 실행할 때 사용합니다. 명명된 프로필에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 명명된 프로필을 참조하십시오. AWS 리전의 목록은 Amazon Web Services 일반 참조의 리전 및 엔드포인트를 참조하십시오.

3. 명령 프롬프트에서 다음 help 명령을 입력하여 설정을 확인합니다.

```
aws translate help
```

Amazon Translate에 대한 간단한 설명과 사용 가능한 명령 목록을 확인해야 합니다.

다음 단계

3단계: 시작하기(콘솔) (p. 7)

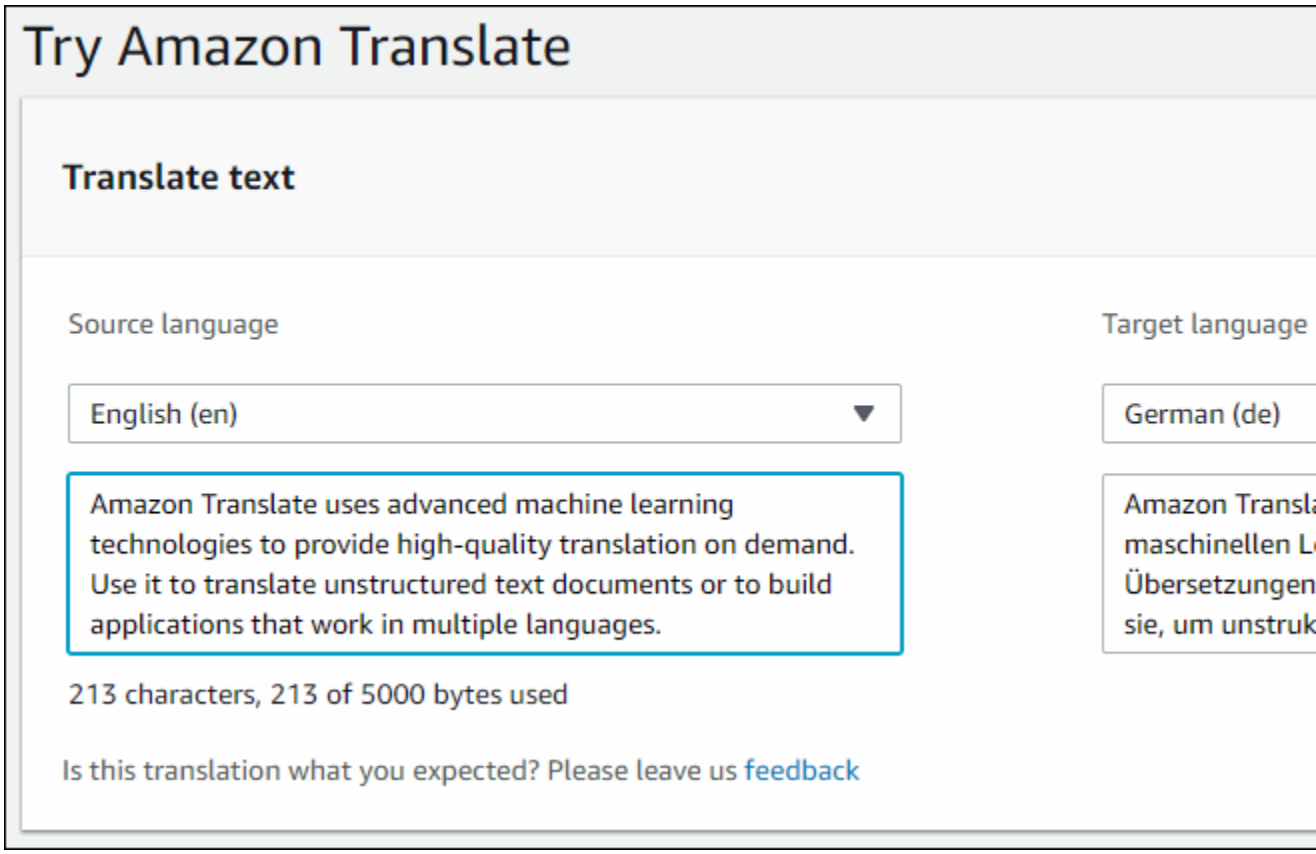
3단계: 시작하기(콘솔)

Amazon Translate를 시작하는 가장 쉬운 방법은 콘솔을 사용하여 텍스트를 번역하는 것입니다. 콘솔을 사용하여 최대 5,000자를 번역할 수 있습니다. Amazon Translate 작동 방식 (p. 3)의 개념과 용어에 대해 아직 모를 경우 먼저 알아두실 것을 권장합니다.

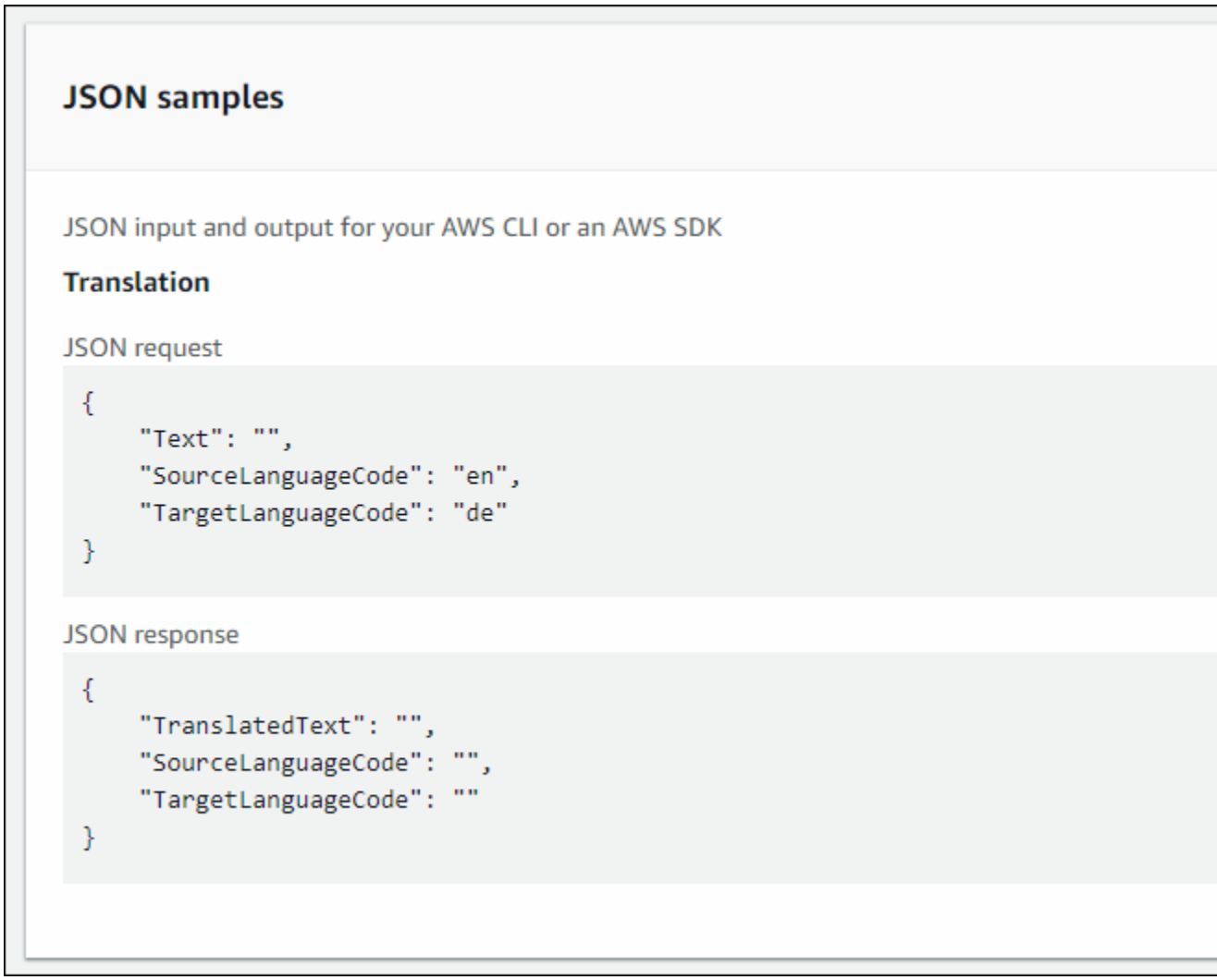
텍스트 번역을 시작하려면 AWS Management 콘솔 에 가서 Amazon Translate 콘솔을 엽니다.

Amazon Translate를 처음 사용하는 경우 Try Amazon Translate를 선택합니다.

Translate text(텍스트 번역)에서 소스 언어와 대상 언어를 선택합니다. 번역하려는 텍스트를 왼쪽 텍스트 상자에 입력합니다. 번역된 텍스트가 오른쪽 텍스트 상자에 나타납니다.



JSON samples(JSON 샘플) 섹션에 TranslateText (p. 71) 작업의 JSON 입력과 출력이 표시됩니다.



다음 단계

4단계: 시작하기(AWS CLI) (p. 8)

4단계: 시작하기(AWS CLI)

다음 연습에서는 AWS 명령줄 인터페이스(AWS CLI)를 사용하여 텍스트를 번역합니다. 다음 연습을 완료하려면 CLI를 능숙하게 사용할 수 있고 텍스트 편집기가 있어야 합니다. 자세한 내용은 2단계: AWS Command Line Interface(AWS CLI) 설정 (p. 6) 섹션을 참조하십시오.

CLI를 사용하여 Amazon Translate로 텍스트를 번역하는 방법은 두 가지가 있습니다. 짧은 텍스트의 경우 번역하려는 텍스트를 `translate-text` 명령의 파라미터로 제공할 수 있습니다. 긴 텍스트의 경우에는 소스 언어, 대상 언어, JSON 파일로 된 텍스트를 제공하면 됩니다.

명령줄에서 Amazon Translate을 사용하려면 서비스 리전과 엔드포인트를 알아야 합니다. 사용 가능한 엔드포인트와 리전 목록은 지침 및 제한 사항 (p. 55) 단원을 참조하십시오.

명령줄을 사용하여 텍스트 번역

다음 예제는 명령줄에서 `translate-text` 작업을 사용하여 텍스트를 번역하는 방법을 보여 줍니다. 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다. 명령줄에 다음을 입력합니다.

```
aws translate translate-text \
            --region region \
            --source-language-code "en" \
            --target-language-code "es" \
            --text "hello, world "
```

응답은 다음과 같은 JSON 형식입니다.

```
{
    "TargetLanguageCode": "es",
    "Text": "Hola, mundo",
    "SourceLanguageCode": "en"
}
```

JSON 파일을 사용하여 텍스트 번역

이 예제는 `translate-text` 작업을 사용하여 JSON 파일의 긴 텍스트 블록을 번역하는 방법을 보여 줍니다. 명령줄에 소스 언어와 대상 언어를 지정할 수 있지만 이 예제에서는 JSON 파일에 언어를 지정합니다.

> **Note**
>
> JSON 파일은 쉽게 읽을 수 있도록 서식이 지정됩니다. 줄바꿈을 제거하려면 `"Text"` 필드의 포맷을 변경합니다.
> 다음은 Unix, Linux, macOS용 형식으로 지정된 예제입니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

JSON 파일을 사용하여 텍스트를 번역하려면

1. 다음 텍스트를 `translate.json`이라는 JSON 파일에 복사합니다.

   ```
   {
       "Text": "Amazon Translate translates documents between languages in
       real time. It uses advanced machine learning technologies
       to provide high-quality real-time translation. Use it to
       translate documents or to build applications that work in
       multiple languages.",
       "SourceLanguageCode": "en",
       "TargetLanguageCode": "fr"
   }
   ```

2. AWS CLI에서 다음 명령을 실행합니다.

   ```
   aws translate translate-text \
               --region region \
               --cli-input-json file://translate.json > translated.json
   ```

   이 명령은 다음 JSON 텍스트를 포함하는 JSON 파일을 출력합니다.

   ```
   {
       "TargetLanguageCode": "fr",
   ```

```
    "Text": "Amazon Translate traduit les documents entre
    les langue en temps réel. Il utilise des technologies
    avancées d'apprentissage de la machine pour fournir
    une traduction en temps réel de haute qualité. Utilisez-le
    pour traduire des documents ou pour créer des applications
    qui fonctionnent en plusieurs langues.",
    "SourceLanguageCode": "en"
}
```

다음 단계

Amazon Translate을 사용하는 다른 방법을 보려면 예제 (p. 20) 단원을 참조하십시오.

5단계: 시작하기(SDK)

다음 예제는 Java 및 Python으로 Amazon Translate TranslateText (p. 71) 작업을 사용하는 방법을 보여 줍니다. 이러한 예제를 통해 `TranslateText` 작업에 대해 알아보고 자체 애플리케이션의 구성 요소로 사용할 수 있습니다.

Java 예제를 실행하려면 AWS SDK for Java를 설치해야 합니다. Java용 SDK를 설치하기 위한 지침은 Java용 AWS SDK 설치를 참조하십시오.

주제

AWS SDK for Java를 사용하여 텍스트 번역

다음 예제는 Java에서 TranslateText (p. 71) 작업을 사용하는 방법을 보여 줍니다. 이 예제를 실행하려면 AWS SDK for Java가 필요합니다. Java용 SDK를 설치하기 위한 지침은 Java용 AWS SDK 설치를 참조하십시오.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.translate.AmazonTranslate;
import com.amazonaws.services.translate.AmazonTranslateClient;
import com.amazonaws.services.translate.model.TranslateTextRequest;
import com.amazonaws.services.translate.model.TranslateTextResult;

public class App {
    private static final String REGION = "region";

    public static void main( String[] args ) {

        // Create credentials using a provider chain. For more information, see
        // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html
        AWSCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonTranslate translate = AmazonTranslateClient.builder()
                .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
```

```
                .withRegion(REGION)
                .build();

        TranslateTextRequest request = new TranslateTextRequest()
                .withText("Hello, world")
                .withSourceLanguageCode("en")
                .withTargetLanguageCode("es");
        TranslateTextResult result  = translate.translateText(request);
        System.out.println(result.getTranslatedText());
    }
}
```

선택한 언어 쌍이 Amazon Translate에서 올바른 조합으로 지원되는 경우에만 원본 언어와 대상 언어를 변경할 수 있습니다. 자세한 내용은 언어 쌍 (p. 1) 단원을 참조하십시오.

AWS SDK for Python (Boto)를 사용하여 텍스트 번역

다음 예제는 Python에서 TranslateText (p. 71) 작업을 사용하는 방법을 보여 줍니다. 이 예제를 실행하려면 먼저 AWS CLI를 통해 Amazon Translate을 설치해야 합니다. 지침은 the section called “2단계: AWS CLI 설정” (p. 6)을 참조하십시오.

```
import boto3

translate = boto3.client(service_name='translate', region_name='region', use_ssl=True)

result = translate.translate_text(Text="Hello, World",
            SourceLanguageCode="en", TargetLanguageCode="de")
print('TranslatedText: ' + result.get('TranslatedText'))
print('SourceLanguageCode: ' + result.get('SourceLanguageCode'))
print('TargetLanguageCode: ' + result.get('TargetLanguageCode'))
```

선택한 언어 쌍이 Amazon Translate에서 올바른 조합으로 지원되는 경우에만 원본 언어와 대상 언어를 변경할 수 있습니다. 자세한 내용은 언어 쌍 (p. 1) 단원을 참조하십시오.

사용자 지정 용어

다음은 Python에서 사용자 지정 용어 작업을 사용하는 방법을 보여 주는 또 하나의 예제입니다.

```
    #!/usr/bin/env python
    # -*- coding: utf-8 -*-

    import boto3

    translate = boto3.client(service_name='translate')

    # The terminology file 'my-first-terminology.csv' has the following contents:
    '''
    en,fr
    Amazon Family,Amazon Famille
    '''

    # Read the terminology from a local file
    with open('/tmp/my-first-terminology.csv', 'rb') as f:
        data = f.read()

    file_data = bytearray(data)

    print("Importing the terminology into Amazon Translate...")
    response = translate.import_terminology(Name='my-first-terminology',
 MergeStrategy='OVERWRITE', TerminologyData={"File": file_data, "Format": 'CSV'})
    print("Terminology imported: "),
```

```
    print(response.get('TerminologyProperties'))
    print("\n")

    print("Getting the imported terminology...")
    response = translate.get_terminology(Name='my-first-terminology',
 TerminologyDataFormat='CSV')
    print("Received terminology: "),
    print(response.get('TerminologyProperties'))
    print("The terminology data file can be downloaded here: " +
 response.get('TerminologyDataLocation').get('Location'))
    print("\n")

    print("Listing the first 10 terminologies for the account...")
    response = translate.list_terminologies(MaxResults=10)
    print("Received terminologies: "),
    print(response.get('TerminologyPropertiesList'))
    print("\n")

    print("Translating 'Amazon Family' from English to French with no terminology...")
    response = translate.translate_text(Text="Amazon Family", SourceLanguageCode="en",
 TargetLanguageCode="fr")
    print("Translated text: " + response.get('TranslatedText'))
    print("\n")

    print("Translating 'Amazon Family' from English to French with the 'my-first-
terminology' terminology...")
    response = translate.translate_text(Text="Amazon Family", TerminologyNames=["my-first-
terminology"], SourceLanguageCode="en", TargetLanguageCode="fr")
    print("Translated text: " + response.get('TranslatedText'))
    print("\n")

    # The terminology file 'my-updated-terminology.csv' has the following contents:
    '''
    en,fr
    Amazon Family,Amazon Famille
    Prime Video, Prime Video
    '''

    # Read the terminology from a local file
    with open('/tmp/my-updated-terminology.csv', 'rb') as f:
        data = f.read()

    file_data = bytearray(data)

    print("Updating the imported terminology in Amazon Translate...")
    response = translate.import_terminology(Name='my-first-terminology',
 MergeStrategy='OVERWRITE', TerminologyData={"File": file_data, "Format": 'CSV'})
    print("Terminology updated: "),
    print(response.get('TerminologyProperties'))
    print("\n")

    print("Translating 'Prime Video' from English to French with no terminology...")
    response = translate.translate_text(Text="Prime Video", SourceLanguageCode="en",
 TargetLanguageCode="fr")
    print("Translated text: " + response.get('TranslatedText'))
    print("\n")

    print("Translating 'Prime Video' from English to French with the 'my-first-terminology'
 terminology...")
    response = translate.translate_text(Text="Prime Video", TerminologyNames=["my-first-
terminology"], SourceLanguageCode="en", TargetLanguageCode="fr")
    print("Translated text: " + response.get('TranslatedText'))
    print("\n")

    print("Cleaning up by deleting 'my-first-terminology'...")
    translate.delete_terminology(Name="my-first-terminology")
```

```
    print("Terminology deleted.")
```

Android용 AWS Mobile SDK를 사용하여 텍스트 번역

Android 애플리케이션에서 Amazon Translate를 사용하여 텍스트를 번역할 수 있습니다.

예제를 구성하려면

1. Android용 AWS Mobile SDK를 설정합니다. 자세한 내용은 AWS 모바일 개발자 가이드의 Android: SDK에 대한 옵션 설정을 참조하십시오.
2. 이 예제를 실행하는 데 필요한 최소한의 권한을 갖는 IAM 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책의 경우 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47)을 참조하십시오. 사용자를 만들거나, 자격 증명을 다운로드하거나, 액세스 키 및 보안 액세스 키를 기록할 수 있습니다.
3. Android Studio를 사용하여 새 프로젝트를 생성합니다.
4. `build.gradle` 파일의 종속성 섹션에 다음을 추가합니다.

```
dependencies {
    implementation 'com.amazonaws:aws-android-sdk-translate:2.6.20'
}
```

5. `AndroidManifest.xml` 파일에 다음 권한을 추가합니다.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

6. 소스 코드를 XCode 프로젝트로 복사합니다.
7. 액세스 키 값과 보안 액세스 키를, 1단계에서 기록한 키로 변경합니다.

코드

다음 코드를 실행하여 예제를 생성합니다.

```
package com.amazonaws.amazontranslatetester;

import android.app.Activity;
import android.util.Log;

import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.handlers.AsyncHandler;
import com.amazonaws.services.translate.AmazonTranslateAsyncClient;
import com.amazonaws.services.translate.model.TranslateTextRequest;
import com.amazonaws.services.translate.model.TranslateTextResult;

public class MainActivity extends Activity {

    private static final String LOG_TAG = MainActivity.class.getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        AWSCredentials awsCredentials = new AWSCredentials() {
            @Override
            public String getAWSAccessKeyId() {
                return "access key";
            }
```

```
            @Override
            public String getAWSSecretKey() {
                return "secret key";
            }
        };

        AmazonTranslateAsyncClient translateAsyncClient = new
 AmazonTranslateAsyncClient(awsCredentials);
        TranslateTextRequest translateTextRequest = new TranslateTextRequest()
                .withText("Hello, world")
                .withSourceLanguageCode("en")
                .withTargetLanguageCode("es");
        translateAsyncClient.translateTextAsync(translateTextRequest, new
 AsyncHandler<TranslateTextRequest, TranslateTextResult>() {
            @Override
            public void onError(Exception e) {
                Log.e(LOG_TAG, "Error occurred in translating the text: " +
 e.getLocalizedMessage());
            }

            @Override
            public void onSuccess(TranslateTextRequest request, TranslateTextResult
 translateTextResult) {
                Log.d(LOG_TAG, "Original Text: " + request.getText());
                Log.d(LOG_TAG, "Translated Text: " +
 translateTextResult.getTranslatedText());
            }
        });
    }
}
```

AWS Mobile SDK for iOS를 사용하여 텍스트 번역

iOS 애플리케이션에서 Amazon Translate를 사용하여 텍스트를 번역할 수 있습니다.

예제를 구성하려면

1. 이 예제를 실행하는 데 필요한 최소한의 권한을 갖는 IAM 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책의 경우 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47)을 참조하십시오. 사용자를 만들거나, 자격 증명을 다운로드하거나, 액세스 키 및 보안 액세스 키를 기록할 수 있습니다.
2. Xcode 버전 8.0 이상을 설치합니다. 최신 버전의 Xcode는 Apple 웹 사이트 https://developer.apple.com/xcode/에서 다운로드할 수 있습니다.
3. Cocoapods 설치. 터미널 창에서 다음 명령을 실행합니다.

   ```
   sudo gem install cocoapods
   ```

4. Xcode를 사용하여 프로젝트를 만듭니다. 그런 다음 터미널 창에서 프로젝트의 `.xcodeproj` 파일이 포함된 디렉터리로 이동하여 다음 명령을 실행합니다.

   ```
   pod init
   ```

5. 핵심 Mobile SDK for iOS 구성 요소를 포드(pod) 파일에 추가합니다.

   ```
   platform :ios, '9.0'
   ```

```
target :'app name' do
   use_frameworks!
   pod 'AWSTranslate', '~> 2.6.19'
   # other pods
end
```

6. 터미널 창에 다음 명령을 실행하여 종속 항목을 설치합니다.

```
pod install --repo-update
```

7. 'pod install'을 실행하면 새로운 워크스페이스 파일이 생성됩니다. Xcode 프로젝트를 닫은 후 *./project_name*.xcworkspace 파일을 사용하여 Xcode 프로젝트를 엽니다. 이제부터는 이 파일을 사용하여 Xcode 프로젝트를 열어야 합니다.

   프로젝트를 연 후 앱을 다시 빌드하여 코드에 호출된 새 라이브러리에서 API를 가져옵니다.

8. 뷰 컨트롤러에 다음 가져오기 문을 추가합니다.

```
import AWSTranslate
```

9. 다음 코드를 XCode 프로젝트에 복사합니다. 액세스 키 값과 보안 키 값을, 1단계에서 기록한 값으로 업데이트합니다.

코드

다음 코드를 실행하여 예제를 생성합니다.

```
var credentialsProvider = AWSStaticCredentialsProvider(accessKey: "access key", secretKey:
 "secret key")

var configuration = AWSServiceConfiguration(region: AWSRegionUSEast1, credentialsProvider:
 credentialsProvider)

AWSServiceManager.default().defaultServiceConfiguration = configuration

let translateClient = AWSTranslate.default()
let translateRequest = AWSTranslateTranslateTextRequest()
translateRequest?.sourceLanguageCode = "en"
translateRequest?.targetLanguageCode = "es"
translateRequest?.text = "Hello World"

let callback: (AWSTranslateTranslateTextResponse?, Error?) -> Void = { (response, error) in
   guard let response = response else {
      print("Got error \(error)")
      return
   }

   if let translatedText = response.translatedText {
      print(translatedText)
   }
}

translateClient.translateText(translateRequest!, completionHandler: callback)
```

사용자 지정 용어

사용자 지정 용어를 번역 요청에 사용하면 브랜드 이름, 캐릭터 이름, 모델 이름 및 기타 고유의 내용을 해당 문맥 및 Amazon Translate 알고리즘의 결정과 상관없이 필요한 방식으로 정확하게 번역할 수 있습니다.

간편하게 용어 파일을 설정하고 Amazon Translate 계정에 연결할 수 있습니다. 텍스트를 번역할 때 사용자 지정 용어도 사용하도록 선택하면 소스 단어의 모든 예가 원하는 대로 번역됩니다.

예를 들어, 다음 경우를 생각해 봅니다. Amazon 패밀리는 Amazon Prime 회원에게 기저귀, 이유식 등에 대한 구독의 최대 20% 할인과 같은 독점 특전을 제공하는 혜택 모음입니다. 프랑스에서는 Amazon Famille라고 합니다. 추가 문맥 없이 Amazon Translate를 사용하여 Amazon 패밀리를 프랑스어로 번역하면 결과적으로 Famille Amazon가 출력됩니다. 이 결과는 정확한 번역이지만 프랑스의 Amazon 팀에게 필요한 결과가 아닙니다. 하지만 "Amazon 패밀리에서 쇼핑해 보셨습니까?"와 같은 문맥을 이 용어에 추가하면 Amazon Translate는 프로그램 이름을 번역할 필요가 없다고 결정하기 때문에 결과는 "Avez-vous déjà fait des achats avec Amazon Family?"가 됩니다. 이 문장은 좋은 번역이지만 여전히 Amazon 팀이 찾고 있는 결과가 아닙니다. 사용자 지정 용어를 사용하면 이와 같은 문제를 해결할 수 있습니다. Amazon 패밀리라는 용어를 Amazon Famille로 번역해야 한다는 것을 보여 주는 항목을 사용자 지정 용어에 추가하면 팀은 이 용어를 문맥과 상관없이 매번 이 방식으로 번역할 수 있습니다. Amazon 패밀리는 이제 Amazon Famille로 번역되고 "Amazon 패밀리에서 쇼핑해 보셨습니까?"는 이제 "Avez-vous déjà fait des achats avec Amazon Famille?"로 번역됩니다

주제

이 기능은 어떻게 작동할까요?

일반적으로 말하면, 번역 요청이 입력되면 Amazon Translate는 소스 문장을 읽고 의미에 맞는 내용 표현을 생성한 다음(다시 말해서 내용을 이해함) 대상 언어로 번역을 생성합니다.

사용자 지정 용어를 번역 요청의 일부로 사용하면 엔진은 최종 결과를 반환하기 전에 용어 파일을 스캔합니다. 엔진이 용어 항목과 소스 텍스트의 문자열 간에 정확한 일치를 식별하면 제안된 번역에서 적절한 문자열을 찾은 다음 해당 문자열을 용어 항목으로 바꿉니다. Amazon 패밀리 예에서 엔진은 처음에 "Avez-vous déjà fait des achats avec Amazon Family?"라는 번역을 생성하지만 중단하고 응답을 제공하기 전에 Amazon Family를 Amazon Famille로 바꿉니다.

사용자 지정 용어 생성

소스 텍스트 및 대상(번역된) 용어가 있는 CSV 파일 또는 TMX 파일을 용어 파일에 사용할 수 있습니다. 각 용어에 단일 소스 텍스트가 사용되지만, 대상 언어와 소스 언어가 사용 가능한 경우 각 언어에 하나씩 여러 대상 용어가 있을 수 있습니다.

CSV(쉼표로 분리된 값)

첫 번째 열에는 소스 텍스트가 포함되고, 다른 열에는 대상 번역이 포함됩니다. 첫 번째 행은 첫 번째 행에 소스 언어가 있고 다른 행에 대상 언어 번역이 있는 언어 코드로 구성됩니다.

en	fr	de	es	
Amazon	Amazon	Amazon	Amazon	

TMX(Translation Memory eXchange)

TMX 파일은 번역 소프트웨어에서 일반적으로 사용되는 XML 유형 파일입니다. 형식은 CSV와 다르지만 내용은 비슷합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<tmx version="1.4">
 <header
    creationtool="XYZTool" creationtoolversion="0"
    datatype="PlainText" segtype="sentence"
    adminlang="en-us" srclang="en"
    o-tmf="test"/>
 <body>
   <tu>
     <tuv xml:lang="en">
       <seg>Amazon</seg>
     </tuv>
     <tuv xml:lang="fr">
       <seg>Amazon</seg>
     </tuv>
     <tuv xml:lang="de">
       <seg>Amazon</seg>
     </tuv>
     <tuv xml:lang="es">
       <seg>Amazon</seg>
     </tuv>
   </tu>
 </body>
</tmx>
```

그러면 이러한 파일은 Amazon Translate 계정에 연결됩니다. 번역 작업이 실행되고 사용자 지정 용어를 사용하도록 선택하면 Amazon Translate는 해당 소스 단어가 나올 때마다 지정된 단어를 사용합니다.

> **Important**
>
> 사용자 지정 용어 내에 소스 단어는 대/소문자를 구분하고 정확하게 일치하지 않는 단어면 사용할 수 없습니다.

호환되는 언어

일부 언어에서는 문장의 문맥에 따라 단어의 형태가 변경되지 않습니다. 이러한 언어를 사용하면 사용자 지정 용어를 적용하면 대체로 전체 번역 품질이 향상됩니다. 하지만 일부 언어에는 광범위한 언어 형태 변화가 있습니다. 이러한 언어에는 이 기능을 적용하지 않는 것이 좋지만, 이 기능을 적용하는 것은 제한되지 않습니다.

다음 표에서는 다양한 언어와 이 기능 사용에 대한 권장 여부를 보여 줍니다.

언어	권장/권장되지 않음
동아시아 언어(예: 중국어, 인도네시아어, 일본어, 한국어, 말레이어)	권장
게르만계 언어(덴마크어, 네덜란드어, 독일어, 영어, 노르웨이어, 스웨덴어)	권장

언어	권장/권장되지 않음
로망스계 언어(프랑스어, 이탈리아어, 스페인어, 포르투갈어)	권장
히브리어	권장
인도이란어 (힌디어, 페르시아어)	권장
슬라브계 언어(체코어, 폴란드어, 러시아어)	권장되지 않음
핀우그리아계 언어(핀란드어)	권장되지 않음
아랍어	권장되지 않음
터키어	권장되지 않음

사용자 지정 용어 사용

TranslateText (p. 71) 작업을 사용하여 텍스트를 번역할 때 사용자 지정 용어를 사용하는 경우 프로세스는 사용자 지정 용어를 사용하지 않고 텍스트를 번역하는 경우와 비슷합니다. 차이점은 사용자 지정 용어와 함께 작업을 호출할 때는 옵션 `TerminologyNames` 파라미터도 포함시킨다는 것 뿐입니다.

예를 들어 `Amazon_Family.csv`라는 다음 용어 파일이 계정과 연결되어 있는 경우

```
en,fr
Amazon Family,Amazon Famille
```

CLI에서 다음을 사용하여 사용자 지정 용어를 호출하고 텍스트를 번역할 수 있습니다.

> **Note**
>
> 이 예제는 Unix, Linux 및 macOS용 형식으로 표시됩니다. Windows의 경우 각 줄의 끝에 있는 백슬래시(\) Unix 연속 문자를 캐럿(^)으로 바꿉니다.

```
aws translate translate-text \
     --region region \
     --source-language-code "en" \
     --target-language-code "fr" \
     --terminology-names "Amazon_Family" \
     --text "Have you ever shopped with Amazon Family?"
```

이 예제는 "Avez-vous déjà fait des achats avec Famille Amazon?"로 직접(하지만 바람직하지 않게) 번역하는 대신 선택한 사용자 지정 용어를 사용하여 이 텍스트를 "Avez-vous déjà fait des achats avec Amazon Famille?"로 번역합니다.

Python을 사용하면 동일한 텍스트가 다음과 같이 보일 수 있습니다.

```
import boto3

translate = boto3.client(service_name='translate')

print("Translating 'Have you ever shopped with Amazon Family?' from English to French with
 the 'Amazon_Family' custom terminology...")
response = translate.translate_text(Text="Have you ever shopped with Amazon Family?",
 TerminologyNames=["Amazon_Family"], SourceLanguageCode="en", TargetLanguageCode="fr")
print("Translated text: " + response.get('TranslatedText'))
```

```
print("\n")
```

사용자 지정 용어와 함께 Amazon Translate 작업을 사용하는 방법에 대한 자세한 내용은 Actions (p. 59)을 참조하십시오.

용어 암호화

Amazon Translate는 모든 데이터를 보호하기 위해 노력하며 사용자 지정 용어도 마찬가지입니다. 생성된 각 사용자 지정 용어는 사용자만 액세스할 수 있도록 암호화됩니다.

세 가지 암호화 옵션을 사용할 수 있습니다.

- AWS 암호화 사용. 이 옵션은 기본 옵션이며 다른 방법을 선택하지 않는 경우 정보를 보호하는 데 사용됩니다.
- 계정과 연결된 암호화 키 사용. 이 옵션을 선택할 경우 콘솔의 메뉴에서 사용할 연결된 암호화 키를 선택할 수 있습니다.
- 계정과 연결되지 않은 암호화 키 사용. 이 옵션을 선택하면 암호화 키의 Amazon 리소스 이름(ARN)을 입력할 수 있는 상자가 콘솔에 표시됩니다.

모범 사례

다음은 사용자 지정 용어를 사용할 때 권장되는 모범 사례입니다.

- 사용자 지정 용어를 최소한으로 유지합니다. 제어하고자 하고 전혀 모호하지 않은 단어만 포함시킵니다. 대체 의미로는 절대 사용하지 않으며 오직 한 가지 방식으로만 번역하려는 단어만 사용합니다. 브랜드 이름 및 제품 이름과 같은 고유 명사로 목록을 제한하는 것이 가장 좋습니다.
- 사용자 지정 용어는 대/소문자를 구분합니다. 대문자인 단어 및 대문자가 아닌 단어가 포함된 두 버전이 필요하면 각 버전에 항목을 포함해야 합니다.
- 동일한 소스 단어에 대해 다른 번역을 포함시키지 마십시오(예: 항목 #1-EN: Amazon, FR: Amazon; 항목 #2-EN: Amazon FR: Amazone).
- 일부 언어에서는 문장의 문맥에 따라 단어의 형태가 변경되지 않습니다. 이러한 언어를 사용하면 사용자 지정 용어를 적용하면 대체로 전체 번역 품질이 향상됩니다. 하지만 일부 언어에는 광범위한 언어 형태 변화가 있습니다. 이러한 언어에는 이 기능을 적용하지 않는 것이 좋지만, 이 기능을 적용하는 것은 제한되지 않습니다. 자세한 내용은 호환되는 언어 (p. 17) 단원을 참조하십시오.

예제

다음 예에서는 Amazon Translate을 사용할 수 있는 방법을 보여 줍니다.

주제

Using Amazon Polly with Amazon Translate

번역된 텍스트를 말로 들으려면 Amazon Polly를 Amazon Translate과 함께 사용합니다. 이 예제에서는 Amazon Translate을 사용하여 텍스트를 번역한 후 Amazon Polly를 사용하여 이 텍스트를 말로 들을 수 있는 웹 페이지를 만들어 보겠습니다. 코드는 다음과 같이 요약할 수 있습니다.

- 웹 페이지를 만들기 위한 CSS 및 HTML.
- Amazon Translate 및 Amazon Polly에 대한 컨트롤러를 만드는 초기화 코드.
- 웹 페이지에서 데이터를 읽고 Amazon Translate을 호출하는 함수.
- 웹 페이지에서 데이터를 읽고 Amazon Polly을 호출하는 함수.
- 웹 페이지를 관리하기 위한 유틸리티 함수.

예제를 구성하려면

1. AWS SDK for JavaScript를 설치하고 구성합니다. SDK for JavaScript 설치 지침은 JavaScript용 SDK 설치를 참조하십시오.
2. 예제 코드를 복사하여 웹 서버의 HTML 파일에 붙여 넣습니다.
3. SDK for JavaScript를 설치한 위치로 `<script>` 태그를 업데이트합니다.
4. 리전과 엔드포인트를 Amazon Translate 및 Amazon Polly 작업을 실행하려는 리전으로 변경합니다. Amazon Translate을 지원하는 리전의 목록은 지침 및 제한 사항 (p. 55) 단원을 참조하십시오. Amazon Polly를 지원하는 리전의 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트를 참조하십시오.
5. 이 예제를 실행하는 데 필요한 최소한의 권한을 갖는 IAM 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책은 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47) 단원과 Amazon Polly 개발자 안내서의 Amazon Polly에서 ID 기반 정책(IAM 정책) 사용 단원을 참조하십시오.
6. 이전 단계에서 만든 IAM 사용자의 액세스 ID와 비밀 키를 제공합니다.

코드

다음은 예제 웹 페이지의 전체 코드입니다. 이 코드를 HTML 파일로 복사하여 웹 서버에서 이 예제를 실행할 수 있습니다.

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Amazon Translate</title>
    <script src="aws-sdk/dist/aws-sdk.js"></script>
</head>

<body>
    <h1 style="text-align: left">Amazon Translate Demo</h1>
    <br/>
    <table class="tg">
        <tr>
            <th align="left">
Source Language Code:
                <select id="sourceLanguageCodeDropdown">
                      <option value="en">en</option>
                      <option value="ar">ar</option>
                      <option value="cs">cs</option>
                      <option value="de">de</option>
                      <option value="es">es</option>
                      <option value="fr">fr</option>
                      <option value="it">it</option>
                      <option value="ja">ja</option>
                      <option value="pt">pt</option>
                      <option value="ru">ru</option>
                      <option value="tr">tr</option>
                      <option value="zh">zh</option>
                      <option value="zh-TW">zh-TW</option>
                </select>
            </th>
            <th align="left">
Target Language Code:
                <select id="targetLanguageCodeDropdown">
                      <option value="en">en</option>
                      <option value="ar">ar</option>
                      <option value="cs">cs</option>
                      <option value="de">de</option>
                      <option value="es">es</option>
                      <option value="fr">fr</option>
                      <option value="it">it</option>
                      <option value="ja">ja</option>
                      <option value="pt">pt</option>
                      <option value="ru">ru</option>
                      <option value="tr">tr</option>
                      <option value="zh">zh</option>
                      <option value="zh-TW">zh-TW</option>
                </select>
            </th>
        </tr>
        <tr>
            <th>
                <textarea id="inputText" name="inputText" rows="10" cols="50"
 placeholder="Text to translate..."></textarea>
            </th>
            <th>
                <textarea id="outputText" name="outputText" rows="10" cols="50"
 placeholder="Translated text..."></textarea>
            </th>
        </tr>
        <tr>
            <th align="left">
                <button type="button" name="translateButton"
 onclick="doTranslate()">Translate</button>
                <button type="button" name="synthesizeButton"
 onclick="doSynthesizeInput()">Synthesize Input Speech</button>
                <button type="button" name="clearButton" onclick="clearInputs()">Clear</
button>
```

```
            </th>
            <th align="left">
                <button type="button" name="synthesizeButton"
 onclick="doSynthesizeOutput()">Synthesize Output Speech</button>
            </th>
        </tr>
    </table>
    <script type="text/javascript">
        // set the focus to the input box
        document.getElementById("inputText").focus();

        /**
        * Change the region and endpoint.
        */
        AWS.config.region = 'region'; // Region

         /**
         * In a production application you should use a secure method of authenticating
 uses, such as the ones
         * described here:
         *   https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-
credentials-browser.html
         *
         * Note that Amazon Translate does not work with Amazon Cognito ## ##.
         *
         * For this example you place the credentials of an IAM user in the HTML page. The
 IAM user associated
         * with these credentials must have permissions to call Amazon Translate. We
 recommend using the following
         * permissions policy and nothing more, as anyone that has access to this HTML page
 will also have access to
         * these hard-coded credentials.
         * {
         *     "Version": "2012-10-17",
         *     "Statement": [
         *         {
         *             "Action": [
         *                 "translate:TranslateText",
         *                 "polly:SynthesizeSpeech"
         *             ],
         *             "Resource": "*",
         *             "Effect": "Allow"
         *         }
         *     ]
         * }
         *
         * For more information about the AWS Credentials object, see:
         *   http://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/Credentials.html
         */
        AWS.config.credentials = new AWS.Credentials("access key", "secret key");

        var translate = new AWS.Translate({region: AWS.config.region});
        var polly = new AWS.Polly();

        function doTranslate() {
            var inputText = document.getElementById('inputText').value;
            if (!inputText) {
                alert("Input text cannot be empty.");
                exit();
            }

            // get the language codes
            var sourceDropdown = document.getElementById("sourceLanguageCodeDropdown");
            var sourceLanguageCode =
 sourceDropdown.options[sourceDropdown.selectedIndex].text;
```

```
            var targetDropdown = document.getElementById("targetLanguageCodeDropdown");
            var targetLanguageCode =
targetDropdown.options[targetDropdown.selectedIndex].text;

            var params = {
                Text: inputText,
                SourceLanguageCode: sourceLanguageCode,
                TargetLanguageCode: targetLanguageCode
            };

            translate.translateText(params, function(err, data) {
                if (err) {
                    console.log(err, err.stack);
                    alert("Error calling Amazon Translate. " + err.message);
                    return;
                }
                if (data) {
                    var outputTextArea = document.getElementById('outputText');
                    outputTextArea.value = data.TranslatedText;
                }
            });
        }

        function doSynthesizeInput() {
            var text = document.getElementById('inputText').value.trim();
            if (!text) {
                return;
            }
            var sourceLanguageCode =
document.getElementById("sourceLanguageCodeDropdown").value;
            doSynthesize(text, sourceLanguageCode);
        }

        function doSynthesizeOutput() {
            var text = document.getElementById('outputText').value.trim();
            if (!text) {
                return;
            }
            var targetLanguageCode =
document.getElementById("targetLanguageCodeDropdown").value;
            doSynthesize(text, targetLanguageCode);
        }

        function doSynthesize(text, languageCode) {
            var voiceId;
            switch (languageCode) {
                case "de":
                    voiceId = "Marlene";
                    break;
                case "en":
                    voiceId = "Joanna";
                    break;
                case "es":
                    voiceId = "Penelope";
                    break;
                case "fr":
                    voiceId = "Celine";
                    break;
                case "pt":
                    voiceId = "Vitoria";
                    break;
                default:
                    voiceId = null;
                    break;
            }
            if (!voiceId) {
```

```
                alert("Speech synthesis unsupported for language code: \"" + languageCode +
 "\"");
                return;
            }
            var params = {
                OutputFormat: "mp3",
                SampleRate: "8000",
                Text: text,
                TextType: "text",
                VoiceId: voiceId
            };
            polly.synthesizeSpeech(params, function(err, data) {
                if (err) {
                    console.log(err, err.stack); // an error occurred
                    alert("Error calling Amazon Polly. " + err.message);
                }
                else {
                    var uInt8Array = new Uint8Array(data.AudioStream);
                    var arrayBuffer = uInt8Array.buffer;
                    var blob = new Blob([arrayBuffer]);
                    var url = URL.createObjectURL(blob);

                    audioElement = new Audio([url]);
                    audioElement.play();
                }
            });
        }

        function clearInputs() {
            document.getElementById('inputText').value = "";
            document.getElementById('outputText').value = "";
            document.getElementById("sourceLanguageCodeDropdown").value = "en";
            document.getElementById("targetLanguageCodeDropdown").value = "en";
        }
    </script>
</body>

</html>
```

Amazon Translate를 사용하여 채팅 채널 번역

채팅 메시지의 실시간 번역에 Amazon Translate를 사용할 수 있습니다. 이 예제는 Twitch 채널을 사용하지만, 이 예제를 다른 채팅 플랫폼, 고객 서비스 상호 작용, 게시판 등과 같은 실시간 스트리밍 텍스트 번역을 위한 시작점으로 활용할 수 있습니다.

이 예제는 실시간 영어 메시지와 실시간 번역을 나란히 보여 주는 웹 페이지를 사용합니다. Amazon Polly로 메시지를 보내서 텍스트를 음성으로 재생할 수 있습니다. 채팅 중인 사람의 말을 이해하려면 그 사람의 사용자 이름을 입력합니다. 그러면 그 사용자의 메시지만 음성으로 재생됩니다.

다음과 같이 코드를 요약할 수 있습니다.

- 웹 페이지를 만들기 위한 CSS 및 HTML.
- Amazon Translate 및 Amazon Polly에 대한 컨트롤러를 만드는 초기화 코드.
- 채팅 메시지를 받으면 실행되는 콜백 함수.
- 채팅 메시지를 보내는 함수.
- Amazon Translate를 호출하여 메시지를 번역하는 함수.
- Amazon Polly를 호출하여 음성을 합성하는 함수.
- 웹 페이지를 관리하기 위한 유틸리티 함수.

예제를 구성하려면

1. AWS SDK for JavaScript를 설치하고 구성합니다. SDK for JavaScript 설치 지침은 JavaScript용 SDK 설치를 참조하십시오.
2. 예제 코드를 복사하여 웹 서버의 HTML 파일에 붙여 넣습니다.
3. SDK for JavaScript를 설치한 위치로 `<script>` 태그를 업데이트합니다.
4. 리전과 엔드포인트를 Amazon Translate 및 Amazon Polly 작업을 실행하려는 리전으로 변경합니다. Amazon Translate을 지원하는 리전의 목록은 지침 및 제한 사항 (p. 55) 단원을 참조하십시오. Amazon Polly를 지원하는 리전의 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트를 참조하십시오.
5. 이 예제를 실행하는 데 필요한 최소한의 권한을 갖는 IAM 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책은 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47) 단원과 Amazon Polly 개발자 안내서의 Amazon Polly에서 ID 기반 정책(IAM 정책) 사용 단원을 참조하십시오.
6. 이전 단계에서 만든 IAM 사용자의 액세스 ID와 비밀 키를 제공합니다.
7. 해당 계정의 Twitch 사용자 이름과 OAuth 토큰을 제공합니다. Twitch 계정은 https://www.twitch.tv에서 만들 수 있습니다. Twitch OAuth 토큰은 https://twitchapps.com/tmi에서 만들 수 있습니다.

```
<!doctype html>
<html lang="en">
<head>
  <title>Amazon Translate</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Latest compiled and minified CSS for Bootstrap -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/
bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/
K68vbdEjh4u" crossorigin="anonymous">

   <!-- Custom CSS -->
   <style>
     .topHeader
     {
       background-color: #6441a4;
       padding: 10px;
       border-bottom: solid 1px #cacaca;
       color: white
     }

     .panelHeading
     {
       background-color: #6441a4 !important;
     }

     .panelBody
     {
      min-height: 450px;  max-height: 450px;overflow-y: scroll;
     }

     body{
        margin-left: 0px;
        margin-right: 0px;
        height: 100%;
      }
   </style>
</head>
<body>
  <div class="container-fluid">
```

```
   <!--Top Header-->
   <div class="row topHeader">
     <div class="col-md-12">
         <h4>Amazon Translate - Artificial Intelligence on AWS - Powerful machine learning
for all Developers and Data Scientists</h4>
     </div>
   </div>

   <!--Status Label-->
   <div class="row">
     <div class="col-md-12">
       <p class="bg-info">
       <div id="connecting-div"></div>
       </p>
     </div>
   </div>

   <div class="row" style="padding: 10px;">
     <div class="col-md-6">
         <div class="form-inline">
           <div class="form-group">
             <input type="text" id="channel" class="form-control" value=""
placeholder="Channel"/>
           </div>
           <div class="form-group">
             <select id="sourceLanguage" class="form-control">
                 <option value="en">en</option>
                 <option value="ar">ar</option>
                 <option value="de" selected="selected">de</option>
                 <option value="es">es</option>
                 <option value="fr">fr</option>
                 <option value="pt">pt</option>
                 <option value="zh">zh</option>
               </select>
           </div>
           <div class="form-group">
             <select id="targetLanguage" class="form-control">
               <option value="en" selected="selected">en</option>
               <option value="ar">ar</option>
               <option value="de">de</option>
               <option value="es">es</option>
               <option value="fr">fr</option>
               <option value="pt">pt</option>
               <option value="zh">zh</option>
             </select>
           </div>
           <div class="form-group">
             <button type="button" class="form-control" id="btn-go"
onclick="connect()">Go</button>
             <button type="button" class="form-control" id="btn-stop"
onclick="location.href='index.html';">Stop</button>
             <span id="status"></span>
           </div>
         </div>
       </div>
       <div class="col-md-6">
           <div class="form-inline">
             <div class="form-group">
                 <input type="checkbox" id="cbSpeak" value="Speak"> Speak Live Translation
                 <input type="text" id="follow" class="form-control" value=""
placeholder="follow"/>
               </div>
           </div>
         </div>
     </div>
```

```
    <!--Chat Boxes-->
    <div class="row">
      <!--Live Chat-->
      <div class="col-md-6">
        <div class="panel panel-primary">
          <div class="panel-heading panelHeading">Live Chat</div>
            <div id="livechatc" class="panel-body panelBody">
              <div class="subscribe" id="livechat"></div>
          </div>
        </div>
      </div>
      <!--Live Chat-->
      <!--Translated Chat-->
      <div class="col-md-6">
        <div class="panel panel-primary">
          <div class="panel-heading panelHeading">Live Translation</div>
            <div id="livetranslationc" class="panel-body panelBody">
              <div class="imageDetected" id="livetranslation"></div>
          </div>
        </div>
      </div>
      <!--Translated Chat-->
    </div>

    <!--Send Message-->
    <div class="row">
        <div class="col-md-11">
            <input type="text" id="message" class="form-control"/>
        </div>
        <div class=" col-md-1">
          <button type="button" class="form-control btn btn-default" id="btn-send"
 onclick="sendMessage()">Send</button>
        </div>
    </div>
  </div>

    <!-- Latest compiled and minified JavaScript -->
    <!-- jQuery first, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
 integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
 crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
 integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
 crossorigin="anonymous"></script>

    <script src="aws-js-sdk/dist/aws-sdk-all.js"></script>
    <script src="http://cdn.tmijs.org/js/1.2.1/tmi.min.js" integrity="sha384-
eE0n7sm1W7DOUI2Xh5I4qSpZTe6hupAO0ovLfqEy0yVJtGRBNfssdmjbJhEYm6Bw"
 crossorigin="anonymous"></script>
    <script>
      cred = {
           twitchUsername: "Twitch user name",
           twitchOAuthToken: "Twitch OAuth token",
           awsAccessKeyId: "access key",
           awsSecretAccessKey: "secret key"
      };

      AWS.config.region = 'region';
      ep = new AWS.Endpoint('endpoint');

      AWS.config.credentials = new AWS.Credentials(cred.awsAccessKeyId,
 cred.awsSecretAccessKey);
      window.translator = new AWS.Translate({endpoint: ep, region: AWS.config.region});

      /*************************Init and Connect to Chat*****************************/
      function connect(){
```

```
  init();

  //Twitch Client
  var options = {
      options: {
        debug: false
      },
      connection: {
        cluster: "aws",
        reconnect: true
      },
      identity: {
        username: cred.twitchUsername,
        password: cred.twitchOAuthToken
      },
      channels: [con.channel]
  };

  window.client = tmi.client(options);

  window.client.connect();

  //Attached Handlers
  window.client.on("chat", onChat);
  window.client.on("connecting", onConnecting);
  window.client.on("connected", onConnected);

  //Disable UI Elements
  document.getElementById("sourceLanguage").disabled = true;
  document.getElementById("targetLanguage").disabled = true;
  document.getElementById("channel").disabled = true;
  document.getElementById("btn-go").disabled = true;
}

function init(){
  //Get UI Controls
  var lc = document.getElementById("livechat");
  var lt = document.getElementById("livetranslation")
  var lcc = document.getElementById("livechatc");
  var ltc = document.getElementById("livetranslationc")
  var cbspeak = document.getElementById("cbSpeak")
  var follow = document.getElementById("follow");
  var sendMessage = document.getElementById("message");

  //Cache values
  con = {
    channel: document.getElementById("channel").value,
    sourceLanguage: document.getElementById("sourceLanguage").value,
    targetLanguage: document.getElementById("targetLanguage").value,
    liveChatUI: lc,
    liveTranslationUI: lt,
    liveChatUIContainer: lcc,
    liveTranslationUIContainer: ltc,
    cbSpeak: cbspeak,
    follow: follow,
    sendMessage: sendMessage
  }

    lc.innerHTML = '';
    lt.innerHTML = '';

    //Speaker
   var voiceId = "Joanna";
   if(con.targetLanguage == "en")
      voiceId = "Joanna";
   else if(con.targetLanguage == "de")
```

```
            voiceId = "Marlene";
       else if(con.targetLanguage == "es")
           voiceId = "Conchita";
       else if(con.targetLanguage == "fr")
          voiceId = "Celine";
       else if(con.targetLanguage == "pt")
          voiceId = "Ines";
       else
          voiceId = "Joanna";
        window.audioPlayer = AudioPlayer(voiceId);
    }
    /*************************Init and Connect to Chat*****************************/

    /*************************Receive and Translate Chat*****************************/
    function onChat (channel, userstate, message, self) {
        // Don't listen to my own messages..
        if (self) return;

        //Translate
        if (message) {
          var username = userstate['username'];

          var params = {
              Text: message,
              SourceLanguageCode: con.sourceLanguage,
              TargetLanguageCode:  con.targetLanguage
          };

          window.translator.translateText(params, function onIncomingMessageTranslate(err,
data) {
             if (err) {
                console.log("Error calling Translate. " + err.message + err.stack);
             }
              if (data) {
                  console.log("M: " + message);
                  console.log("T: " + data.TranslatedText);

                  //Print original message in chat UI
                  con.liveChatUI.innerHTML += '<strong>' + username + '</strong>: ' +
message + '<br>';

                  //Print translation in translation UI
                  con.liveTranslationUI.innerHTML += '<strong>' + username + '</strong>: '
+ data.TranslatedText + '<br>';

                  //If speak translation in enabled, speak translated message
                  if(con.cbSpeak.checked){
                    if(con.follow.value == "" || username == con.follow.value)
                      audioPlayer.Speak(username + " says " + data.TranslatedText);
                  }

                  //Scroll chat and translated UI to bottom to keep focus on latest
messages
                  con.liveChatUIContainer.scrollTop = con.liveChatUIContainer.scrollHeight;
                  con.liveTranslationUIContainer.scrollTop =
con.liveTranslationUIContainer.scrollHeight;
               }
          });
        }
    }
    /*************************Receive and Translate Chat*****************************/

    /*************************Client Connecting*****************************/
    function onConnecting (address, port) {
        document.getElementById("status").innerHTML = " [ Connecting...]"
    }
```

```
    function onConnected (address, port) {
        document.getElementById("status").innerHTML = " [ Connected ]"
        window.audioPlayer.Speak("Connected to channel " + con.channel + ". You should now
be getting live chat messages.");
    }
    /*************************Client Connecting*****************************/

    /*************************Send Message*****************************/
    function sendMessage(){
         if(con.sendMessage.value){
           message = con.sendMessage.value;
           var params = {
               Text: con.sendMessage.value,
               SourceLanguageCode: con.targetLanguage,
               TargetLanguageCode:  con.sourceLanguage
           };

           window.translator.translateText(params, function onSendMessageTranslate(err,
data) {
                 if (err) {
                    console.log("Error calling Translate. " + err.message + err.stack);
                 }
                 if (data) {
                     console.log("M: " + message);
                     console.log("T: " + data.TranslatedText);

                     //Send message to chat
                     window.client.action(con.channel, data.TranslatedText);

                     //Clear send message UI
                     con.sendMessage.value = "";

                     //Print original message in Translated UI
                     con.liveTranslationUI.innerHTML += '<strong> ME: </strong>: ' +
message + '<br>';

                     //Print translated message in Chat UI
                     con.liveChatUI.innerHTML += '<strong> ME: </strong>: ' +
data.TranslatedText + '<br>';

                     //Scroll chat and translated UI to bottom to keep focus on latest
messages
                     con.liveChatUIContainer.scrollTop =
con.liveChatUIContainer.scrollHeight;
                     con.liveTranslationUIContainer.scrollTop =
con.liveTranslationUIContainer.scrollHeight;
                 }
           });
         }
    }
    /*************************Send Message*****************************/

    /*************************Audio player*****************************/
    function AudioPlayer(voiceId) {
        var audioPlayer = document.createElement('audio');
        audioPlayer.setAttribute("id", "audioPlayer");
        document.body.appendChild(audioPlayer);

        var isSpeaking = false;

        var speaker = {
            self: this,
            playlist:[],

            Speak: function (text) {
```

```
                //If currently speaking a message, add new message to the playlist
                if (isSpeaking) {
                    this.playlist.push(text);
                } else {
                    speakTextMessage(text).then(speakNextTextMessage)
                }
            }
        }

        // Speak text message
        function speakTextMessage(text) {
            return new Promise(function (resolve, reject) {
                isSpeaking = true;
                getAudioStream(text).then(playAudioStream).then(resolve);
            });
        }

        // Speak next message in the list
        function speakNextTextMessage() {
            var pl = speaker.playlist;
            if (pl.length > 0) {
                var txt = pl[0];
                pl.splice(0, 1);
                speakTextMessage(txt).then(speakNextTextMessage);
            }
        }

        // Get synthesized speech from Amazon polly
        function getAudioStream(textMessage) {
            return new Promise(function (resolve, reject) {
                var polly = new AWS.Polly();
                var params = {
                    OutputFormat: 'mp3',
                    Text: textMessage,
                    VoiceId: voiceId
                }
                polly.synthesizeSpeech(params, function (err, data) {
                    if (err)
                        reject(err);
                    else
                        resolve(data.AudioStream);
                });
            });
        }

        // Play audio stream
        function playAudioStream(audioStream) {
            return new Promise(function (resolve, reject) {
                var uInt8Array = new Uint8Array(audioStream);
                var arrayBuffer = uInt8Array.buffer;
                var blob = new Blob([arrayBuffer]);

                var url = URL.createObjectURL(blob);
                audioPlayer.src = url;
                audioPlayer.addEventListener("ended", function () {
                    isSpeaking = false;
                    resolve();
                });
                audioPlayer.play();
            });
        }

        return speaker;
    }
    /**************************Audio player*****************************/
</script>
```

```
</body>
</html>
```

Using Amazon Translate with Amazon DynamoDB

이 예제는 제품 후기를 번역하여 Amazon DynamoDB에 저장하는 방법을 보여 줍니다. 나중에 동일한 후기를 요청하면 Amazon Translate를 통해 다시 번역할 필요 없이 DynamoDB에서 번역을 반환합니다.

이 예제에서는 다음과 같이 합니다.

- AWS CloudFormation을 사용하여 번역을 저장하는 DynamoDB 테이블을 만들고, TranslateText (p. 71) 작업을 호출하는 Lambda 함수를 만듭니다.
- AWS Lambda 콘솔을 사용하여 함수를 테스트합니다.

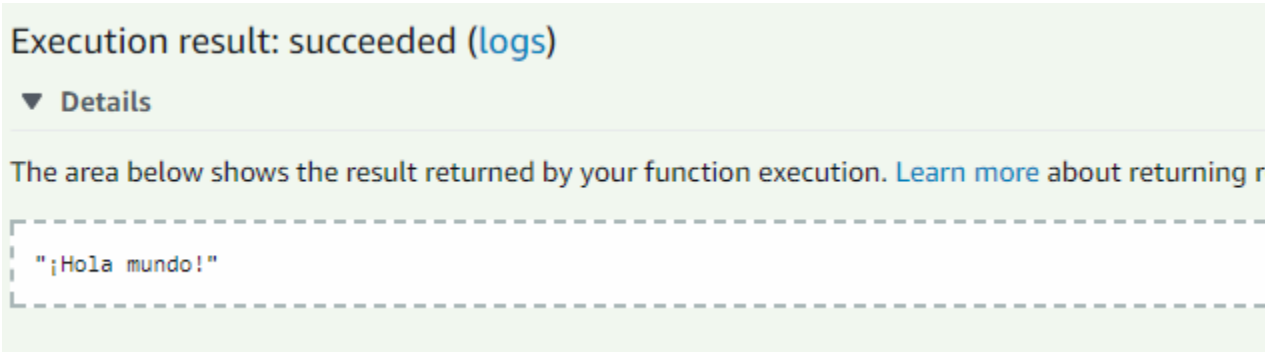
예제를 실행하려면

1. Python Lambda 함수 (p. 33)에서 `example.py`의 내용을 복사하여 `example.py`라는 파일에 붙여 넣습니다. `example.py`는 TranslateText (p. 71) 작업을 호출하는 Lambda 함수입니다. 파일을 `example.zip`이라는 zip 아카이브로 압축합니다. 함수를 실행하려는 리전과 동일한 AWS 리전에 있는 S3 버킷에 파일을 저장합니다.
2. `template.yaml`라는 이름의 새로운 파일을 만듭니다. AWS CloudFormation 템플릿 (p. 34)에서 찾을 수 있는 AWS CloudFormation 템플릿 코드를 파일로 복사합니다. AWS CloudFormation는 이 템플릿을 사용하여 샘플 애플리케이션용 리소스를 생성합니다. `BUCKET_NAME`을 `example.zip`이 저장된 S3 버킷의 이름으로 변경합니다. 파일을 로컬 디렉터리에 저장합니다.
3. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
4. [Create new stack]을 선택합니다.
5. [Upload a template to Amazon S3]을 선택한 후 [Choose file]을 선택합니다. 2단계에서 만든 `template.yaml`을 선택한 후 [Next]를 선택합니다.
6. 스택 이름을 입력한 후 [Next]를 선택합니다.
7. [Options] 페이지에서 [Next]를 선택합니다.
8. I acknowledge that AWS CloudFormation might create IAM resources와 I acknowledge that AWS CloudFormation might create IAM resources with custom names를 선택합니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 AWS Identity and Access Management를 사용하여 액세스 제어를 참조하십시오.
9. [Create Change Set]를 선택합니다.
10. AWS CloudFormation이 변경 세트를 생성하면 Execute를 선택합니다. AWS CloudFormation이 스택을 생성할 때까지 기다립니다.
11. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
12. 새 함수를 선택합니다. 이름이 `TestTranslate-ReviewTranslate`로 시작하는 함수입니다.
13. 함수 세부 정보 페이지에서 [Test]를 선택합니다.
14. 이름에 **`TestTranslate`**를 입력합니다. [Configure test events] 서 JSON을 다음으로 바꿉니다.

```
{
   "review": "hello world",
   "target_language": "es",
   "source_language": "en",
   "review_id": "1"
}
```

Create를 선택합니다.

15. [TestTranslate]를 선택했는지 확인한 후 [Test]를 선택합니다. 테스트가 완료되면 다음 메시지가 표시됩니다.



예제 코드

다음 코드를 실행하여 예제를 생성합니다.

Python Lambda 함수

다음은 Python Lambda 함수의 내용입니다. Lambda 함수는 `TranlateText` 작업을 호출하여 후기와 출발 언어 및 도착 언어를 전달하여 번역된 후기를 받습니다. 이 파일을 `example.py`로 저장한 후 `example.zip`이라는 .zip 아카이브로 압축합니다. 예제를 실행하는 리전과 동일한 리전의 S3 버킷에 파일을 저장합니다.

```
import logging
import json
import boto3
import os

translate = boto3.client('translate')
dynamodb = boto3.client('dynamodb')
firehose = boto3.client('firehose')

TABLE_NAME = os.getenv('TABLE_NAME')

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):

    logger.info(event)

    if 'source_language' in event and 'target_language' in event and 'review' in event and
 'review_id' in event:
        review_id = event['review_id']
        source_language = event['source_language']
        target_language = event['target_language']
        review = event['review']

        try:
            # The Lambda function queries the Amazon DynamoDB table to check whether
            # the review has already been translated. If the translated review
            # is already stored in Amazon DynamoDB, the function returns it.
            response = dynamodb.get_item(
                TableName=TABLE_NAME,
                Key={
                    'review_id': {
```

```
                        'N': review_id,
                    },
                    'language': {
                        'S': target_language,
                    },
                }
            )
            logger.info(response)
            if 'Item' in response:
                return response['Item']['review']['S']
        except Exception as e:
            logger.error(response)
            raise Exception("[ErrorMessage]: " + str(e))

        try:
            # The Lambda function calls the TranslateText operation and passes the
            # review, the source language, and the target language to get the
            # translated review.
            result = translate.translate_text(Text=review,
 SourceLanguageCode=source_language, TargetLanguageCode=target_language)
            logging.info("Translation output: " + str(result))
        except Exception as e:
            logger.error(response)
            raise Exception("[ErrorMessage]: " + str(e))

        try:
            # After the review is translated, the function stores it using
            # the Amazon DynamoDB putItem operation. Subsequent requests
            # for this translated review are returned from Amazon DynamoDB.
            response = dynamodb.put_item(
            TableName=TABLE_NAME,
            Item={
                'review_id': {
                    'N': review_id,
                },
                'language': {
                    'S': target_language,
                },
                'review': {
                    'S': result.get('TranslatedText')
                }
                }
            )
            logger.info(response)
        except Exception as e:
                logger.error(e)
                raise Exception("[ErrorMessage]: " + str(e))
        return result.get('TranslatedText')
    else:
        logger.error(e)
        raise Exception("[ErrorMessage]: Invalid input ")
```

AWS CloudFormation 템플릿

다음은 AWS CloudFormation에서 Lambda 함수와 DynamoDB 테이블을 생성 및 구성할 때 사용하는 템플릿 파일입니다. 예제의 AWS CloudFormation 스택을 생성할 때 이 파일을 사용하십시오. `BUCKET_NAME`을, `example.zip` 파일을 포함하는 S3 버킷의 이름으로 업데이트한 후 로컬 디렉터리에 `template.yaml`로 저장합니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Resources:  ReviewTranslate:
    Type: 'AWS::Serverless::Function'
```

```
  Properties:
      Handler: example.lambda_handler
      Runtime: python2.7
      CodeUri:
        Bucket: BUCKET_NAME
        Key: example.zip
      Policies:
       - AWSLambdaFullAccess
       - TranslateReadOnly
      Environment:
        Variables:
          TABLE_NAME: !Ref ReviewTable
      Tracing: "Active"
ReviewTable:
  Type: 'AWS::DynamoDB::Table'
  Properties:
      AttributeDefinitions:
          - AttributeName: "review_id"
            AttributeType: "N"
          - AttributeName: "language"
            AttributeType: "S"
      KeySchema:
          - AttributeName: "review_id"
            KeyType: "HASH"
          - AttributeName: "language"
            KeyType: "RANGE"
      ProvisionedThroughput:
          ReadCapacityUnits: 5
          WriteCapacityUnits: 5
```

Amazon Translate를 사용하여 웹 페이지 번역

Amazon Translate를 사용하여 웹 페이지 콘텐츠를 번역할 수 있습니다. 다음 Java 프로그램에서는 지정된 웹 페이지를 영어에서 스페인어로 번역하고 번역된 결과가 포함된 HTML 파일을 생성합니다. 프로그램에는 두 개의 함수가 있습니다.

- 소스 웹 페이지에서 데이터를 읽는 함수와 HTML 요소와 데이터를 분리한 다음 요소를 번역하기 위해 두 번째 함수를 호출합니다. 설명서 마지막에 HTML 파일로 결과를 작성합니다.
- HTML 요소의 콘텐츠를 번역하기 위해 Amazon Translate 서비스를 호출하는 함수입니다.

이 예제는 중첩 요소 없이 간단한 HTML 페이지에서 실행됩니다.

예제를 구성하려면

1. AWS SDK for Java를 설치하고 구성합니다. Java용 SDK를 설치하는 방법은 AWS SDK for Java 설치하기 단원을 참조하십시오.
2. jsoup Java HTML 파서를 설치합니다. 지침은 jsoup를 참조하십시오.
3. 이 예제를 실행하는 데 필요한 최소한의 권한을 갖는 IAM 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책의 경우 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47)을 참조하십시오.
4. 필요한 자격 증명을 설치하여 샘플을 실행합니다. 지침의 경우 AWS SDK for Java 개발자 안내서의 개발을 위한 자격 증명 및 리전 AWS 설정하기를 참조하십시오.
5. Java IDE에 새 프로젝트를 생성하고 소스 코드를 복사합니다.
6. 리전과 엔드포인트를 Amazon Translate 작업을 실행하려는 리전으로 변경합니다. Amazon Translate을 지원하는 리전의 목록은 지침 및 제한 사항 (p. 55) 단원을 참조하십시오.

```
package com.amazonaws.translateweb;

import com.amazonaws.auth.AWSCredentialsProviderChain;
import com.amazonaws.auth.EnvironmentVariableCredentialsProvider;
import com.amazonaws.auth.SystemPropertiesCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.translate.AmazonTranslate;
import com.amazonaws.services.translate.AmazonTranslateClient;
import com.amazonaws.services.translate.model.TranslateTextRequest;
import com.amazonaws.services.translate.model.TranslateTextResult;
import com.amazonaws.AmazonServiceException;

import java.io.IOException;
import java.io.PrintWriter;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;


public class TranslateWebPage {

   public static void main(String[] args) throws InterruptedException {

   // Define the URL of the HTML content to translate
   String url = "http://example.com/source.html";

   // Create credentials using a provider chain that will evaluate in order;
   // a) Any Java system properties
   // b) Any environment variables
   // c) Any profile file
   AWSCredentialsProviderChain DefaultAWSCredentialsProviderChain = new
 AWSCredentialsProviderChain(
         new SystemPropertiesCredentialsProvider(),
         new EnvironmentVariableCredentialsProvider(),
         new ProfileCredentialsProvider()
      );

   // Create an endpoint configuration for the Translate service
   AwsClientBuilder.EndpointConfiguration endpointConfiguration = new
 AwsClientBuilder.EndpointConfiguration(
      "endpoint", "region");

   // Create a client for the Translate service
   AmazonTranslate translate = AmazonTranslateClient.builder()
      .withCredentials(DefaultAWSCredentialsProviderChain)
      .withEndpointConfiguration(endpointConfiguration).build();


   // Record the beginning of translating the HTML content at the url
   System.out.println("Translating URL: " + url);

   // Create an empty HTML document to store the parsed data
   Document doc;

   try {
      // Retrieve the HTML located at the URL
      doc = Jsoup.connect(url).get();

      // Select all of the elements in the HTML
      Elements eles = doc.select("*");
```

```
      // For each element
      for (Element ele : eles) {

          // Translate the element
          translateElement(ele, translate);

          // If you encounter service throttling when translating large web
          // pages, you can request a service limit increase. For details,
          // see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-
limits/,
          // or you can throttle your requests by inserting a sleep statement.
          // Thread.sleep(1000);
      }

      // Configure an output file for the translated HTML
      String fname = "output HTML file name";
      PrintWriter pw = new PrintWriter(fname, "UTF-8");

      // Write our translated HTML to the output file
      pw.println(doc);
      pw.close();

      // Record that the file has been saved
      System.out.println("Saved file "+fname);


      // Catch any exceptions in retrieving the HTML
   } catch (IOException e1) {
      e1.printStackTrace();
   }
}

   // This function is used to translate each individual element
   public static void translateElement(Element ele, AmazonTranslate translate) {

      // Check if the element has any text
      if (!ele.ownText().isEmpty()) {

         // Retrieve the text of the HTML element
         String text = ele.ownText();

         // Now translate the element's text
         try {

            // Translate from English to Spanish
            TranslateTextRequest request = new TranslateTextRequest()
                .withText(text)
                .withSourceLanguageCode("en")
                .withTargetLanguageCode("es");

            // Retrieve the result
            TranslateTextResult result  = translate.translateText(request);

            // Record the original and translated text
            System.out.println("Original text: " + text + " - Translated text: "+
 result.getTranslatedText());

            // Update the HTML element with the translated text
            ele.text(result.getTranslatedText());

            // Catch any translation errors
         } catch (AmazonServiceException e) {
                   System.err.println(e.getErrorMessage());
                   System.exit(1);
         }
      } else {
```

```
        // We have found a non-text HTML element. No action required.
      }
    }
}
```

Amazon Translate 이용한 큰 문서 번역

큰 문서를 작은 부분으로 분리하여 총 문서 크기를 문서 크기 한도 이내로 유지할 수 있습니다. 문서 크기 한도에 대한 자세한 내용은 서비스 제한 (p. 55) 단원을 참조하십시오. 다음 Java 프로그램은 긴 텍스트 문서를 개별 문장으로 분리한 후 소스 언어의 각 문장을 대상 언어로 번역합니다. 프로그램은 다음 두 섹션으로 구성됩니다.

- 원본 문자열을 개별 문장으로 분리하는 `SentenceSegmenter` 클래스. 이 샘플에서는 Java `BreakIterator` 클래스를 사용합니다.

- `main` 함수는 원본 문자열의 각 문장에 `Translate` 연산을 호출합니다. `main` 함수는 Amazon Translate를 이용한 인증도 처리합니다.

예제를 구성하려면

1. AWS SDK for Java를 설치하고 구성합니다. Java용 SDK를 설치하는 방법은 AWS SDK for Java 설치하기 단원을 참조하십시오.
2. 이 예제를 실행하는 데 필요한 최소한의 권한을 갖는 IAM 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책의 경우 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47)을 참조하십시오.
3. 필요한 자격 증명을 설치하여 샘플을 실행합니다. 지침의 경우 AWS SDK for Java 개발자 안내서의 개발을 위한 자격 증명 및 리전 AWS 설정하기를 참조하십시오.
4. Java IDE에 새 프로젝트를 생성하고 소스 코드를 복사합니다.
5. 리전을 Amazon Translate 작업을 실행하려는 리전으로 변경합니다. Amazon Translate을 지원하는 리전의 목록은 지침 및 제한 사항 (p. 55) 단원을 참조하십시오.
6. 소스 언어와 대상 언어를 번역하려는 언어로 변경합니다.
7. 샘플을 실행하여 표준 출력 상의 번역문을 확인하십시오.

```
import com.amazonaws.auth.AWSCredentialsProviderChain;
import com.amazonaws.auth.EnvironmentVariableCredentialsProvider;
import com.amazonaws.auth.SystemPropertiesCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.translate.AmazonTranslate;
import com.amazonaws.services.translate.AmazonTranslateClient;
import com.amazonaws.services.translate.model.TranslateTextRequest;
import com.amazonaws.services.translate.model.TranslateTextResult;
import java.text.BreakIterator;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class MultiSentenceTranslator {

    public static void main(String[] args) {
        // Define the text to be translated here
        String region = "region";
        String text = "Text to be translated";
```

```
        String sourceLang = "source language";
        String targetLang = "target language";

        // Break text into sentences
        SentenceSegmenter sentenceSegmenter = new SentenceSegmenter();
        List<String> sentences = new ArrayList<>();
        try {
            sentences = sentenceSegmenter.segment(text, sourceLang);
        } catch (Exception e) {
            System.out.println(e);
            System.exit(1);
        }

        // Create credentials using a provider chain that will evaluate in order;
        // a) Any Java system properties
        // b) Any environment variables
        // c) Any profile file
        AWSCredentialsProviderChain DefaultAWSCredentialsProviderChain = new
 AWSCredentialsProviderChain(
                new SystemPropertiesCredentialsProvider(),
                new EnvironmentVariableCredentialsProvider(),
                new ProfileCredentialsProvider()
        );

        // Create an Amazon Translate client
        AmazonTranslate translate = AmazonTranslateClient.builder()
                .withCredentials(DefaultAWSCredentialsProviderChain)
                .withRegion(region)
                .build();

        // Translate sentences and print the results to stdout
        for (String sentence : sentences) {
            TranslateTextRequest request = new TranslateTextRequest()
                    .withText(sentence)
                    .withSourceLanguageCode(sourceLang)
                    .withTargetLanguageCode(targetLang);
            TranslateTextResult result = translate.translateText(request);
            System.out.println("Original text: " + sentence);
            System.out.println("Translated text: " + result.getTranslatedText());
        }
    }

}

class SentenceSegmenter {
    public List<String> segment(final String text, final String lang) throws Exception {
        List<String> res = new ArrayList<>();
        BreakIterator sentenceIterator = BreakIterator.getSentenceInstance(new
 Locale(lang));
        sentenceIterator.setText(text);
        int prevBoundary = sentenceIterator.first();
        int curBoundary = sentenceIterator.next();
        while (curBoundary != BreakIterator.DONE) {
            String sentence = text.substring(prevBoundary, curBoundary);
            res.add(sentence);
            prevBoundary = curBoundary;
            curBoundary = sentenceIterator.next();
        }
        return res;
    }

}
```

Amazon Translate에서 서명 버전 4 사용

이 예제 Python 프로그램은 서명 버전 4를 사용하여 Amazon Translate 요청에 인증 정보를 추가하는 방법을 보여 줍니다. 이 예제는 POST 요청을 생성하고, 요청 본문(페이로드)에 번역할 텍스트를 포함하는 JSON 구조를 만들고, Authorization 헤더의 인증 정보를 전달합니다. 서명 버전 4 사용에 대한 자세한 내용은 Amazon Web Services 일반 참조의 서명 버전 4 서명 프로세스를 참조하십시오.

설정

예제를 실행하려면 다음 단계를 수행합니다.

1. AWS Command Line Interface(AWS CLI)를 설치합니다. AWS SDK for Python (Boto)은 AWS CLI를 설치할 때 포함됩니다. 지침은 2단계: AWS Command Line Interface(AWS CLI) 설정 (p. 6)을 참조하십시오.
2. 이 예제를 실행하는 데 필요한 최소한의 권한 정책을 갖는 AWS Identity and Access Management(IAM) 사용자를 만듭니다. IAM 사용자 생성에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하십시오. 필요한 권한 정책의 경우 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47)을 참조하십시오. 사용자 액세스 키 ID와 보안 액세스 키를 기록합니다.
3. `AWS_ACCESS_KEY and AWS_SECRET_ACCESS_KEY`라는 환경 변수에 액세스 키 ID와 보안 액세스 키를 각각 넣습니다. 자격 증명을 코드에 포함하지 않는 것이 가장 좋은 방법입니다.
4. 컴퓨터에 새 파일을 만들고 예제용 코드(다음 단원 참조)를 복사하여 파일에 붙여 넣은 후 `.py` 확장명으로 파일을 저장합니다.
5. 코드에서 `region`을, Amazon Translate `TranslateText` 작업을 실행하려는 AWS 리전 이름으로 바꿉니다. 지원되는 리전 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트를 참조하십시오.

코드

다음은 예제 Python 프로그램의 전체 코드입니다.

엔드포인트 URL 및 요청 본문과 같은 요청 값을 생성한 후 이 코드는 다음을 수행합니다.

1. Amazon Translate `TranslateText` 작업에 대한 정식 요청을 생성합니다.
2. 서명을 생성하기 해시를 생성하려는 문자열을 생성합니다.
3. 서명을 계산합니다.
4. 요청 헤더에 서명을 추가합니다.
5. `TranslateText` 작업에 요청을 전송합니다.

컴퓨터에서 예제를 실행하려면 Python 파일로 코드를 복사합니다.

```
# AWS Version 4 signing example

# Translate API (TranslateText)

# For more information about using Signature Version 4, see http://docs.aws.amazon.com/
general/latest/gr/sigv4_signing.html.
# This example makes a POST request to Amazon Translate and
# passes the text to translate JSON in the body (payload)
# of the request. Authentication information is passed in an
# Authorization header.
import sys, os, base64, datetime, hashlib, hmac
```

```
import requests # pip install requests

# ************* REQUEST VALUES *************
method = 'POST'
service = 'translate'
region = 'region'
host = service + '.' + region + '.amazonaws.com'
endpoint = 'https://' + host + '/'

# POST requests use a content type header. For Amazon Translate,
# the content is JSON.
content_type = 'application/x-amz-json-1.1'
# Amazon Translate requires an x-amz-target header that has this format:
#     AWSShineFrontendService_20170701.<operationName>.
amz_target = 'AWSShineFrontendService_20170701.TranslateText'

# Pass request parameters for the TranslateText operation in a JSON block.
request_parameters =  '{'
request_parameters +=  '"Text": "Hello world.",'
request_parameters +=  '"SourceLanguageCode": "en",'
request_parameters +=  '"TargetLanguageCode": "de"'
request_parameters +=  '}'

# The following functions derive keys for the request. For more information, see
# http://docs.aws.amazon.com/general/latest/gr/signature-v4-examples.html#signature-v4-
examples-python.
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()

def getSignatureKey(key, date_stamp, regionName, serviceName):
    kDate = sign(('AWS4' + key).encode('utf-8'), date_stamp)
    kRegion = sign(kDate, regionName)
    kService = sign(kRegion, serviceName)
    kSigning = sign(kService, 'aws4_request')
    return kSigning

# Python can read the AWS access key from environment variables or the configuration file.
# In this example, keys are stored in environment variables. As a best practice, do not
# embed credentials in code.
access_key = os.environ.get('AWS_ACCESS_KEY_ID')
secret_key = os.environ.get('AWS_SECRET_ACCESS_KEY')
if access_key is None or secret_key is None:
    print 'No access key is available.'
    sys.exit()

# Create a timestamp for headers and the credential string.
t = datetime.datetime.utcnow()
amz_date = t.strftime('%Y%m%dT%H%M%SZ')
date_stamp = t.strftime('%Y%m%d') # The date without time is used in the credential scope.


# ************* TASK 1: CREATE A CANONICAL REQUEST *************
# For information about creating a canonical request, see http://docs.aws.amazon.com/
general/latest/gr/sigv4-create-canonical-request.html.

# Step 1: Define the verb (GET, POST, etc.), which you have already done.

# Step 2: Create a canonical URI. A canonical URI is the part of the URI from domain to
 query.
# string (use '/' if no path)
canonical_uri = '/'

## Step 3: Create the canonical query string. In this example, request
# parameters are passed in the body of the request and the query string
# is blank.
canonical_querystring = ''
```

```
# Step 4: Create the canonical headers. Header names must be trimmed,
# lowercase, and sorted in code point order from low to high.
# Note the trailing \n.
canonical_headers = 'content-type:' + content_type + '\n' + 'host:' + host + '\n' + 'x-amz-
date:' + amz_date + '\n' + 'x-amz-target:' + amz_target + '\n'

# Step 5: Create the list of signed headers by listing the headers
# in the canonical_headers list, delimited with ";" and in alphabetical order.
# Note: The request can include any headers. Canonical_headers and
# signed_headers should contain headers to include in the hash of the
# request. "Host" and "x-amz-date" headers are always required.
# For Amazon Translate, content-type and x-amz-target are also required.
signed_headers = 'content-type;host;x-amz-date;x-amz-target'

# Step 6: Create the payload hash. In this example, the request_parameters
# variable contains the JSON request parameters.
payload_hash = hashlib.sha256(request_parameters).hexdigest()

# Step 7: Combine the elements to create a canonical request.
canonical_request = method + '\n' + canonical_uri + '\n' + canonical_querystring + '\n' +
 canonical_headers + '\n' + signed_headers + '\n' + payload_hash


# ************* TASK 2: CREATE THE STRING TO SIGN*************
# Set the algorithm variable to match the hashing algorithm that you use, either SHA-256
 (recommended) or SHA-1.
#
algorithm = 'AWS4-HMAC-SHA256'
credential_scope = date_stamp + '/' + region + '/' + service + '/' + 'aws4_request'
string_to_sign = algorithm + '\n' +  amz_date + '\n' +  credential_scope + '\n' +
 hashlib.sha256(canonical_request).hexdigest()


# ************* TASK 3: CALCULATE THE SIGNATURE *************
# Create the signing key using the getSignaturKey function defined above.
signing_key = getSignatureKey(secret_key, date_stamp, region, service)

# Sign the string_to_sign using the signing_key.
signature = hmac.new(signing_key, (string_to_sign).encode('utf-8'),
 hashlib.sha256).hexdigest()


# ************* TASK 4: ADD SIGNING INFORMATION TO THE REQUEST *************
# Put the signature information in a header named Authorization.
authorization_header = algorithm + ' ' + 'Credential=' + access_key + '/' +
 credential_scope + ', ' +  'SignedHeaders=' + signed_headers + ', ' + 'Signature=' +
 signature

# For Amazon Translate, the request can include any headers, but it must include "host,"
 "x-amz-date,"
# "x-amz-target," "content-type," and "Authorization" headers. Except for the authorization
# header, the headers must be included in the canonical_headers and signed_headers values,
 as
# noted earlier. Header order is not significant.
# Note: The Python 'requests' library automatically adds the 'host' header.
headers = {'Content-Type':content_type,
           'X-Amz-Date':amz_date,
           'X-Amz-Target':amz_target,
           'Authorization':authorization_header}


# ************* TASK 5: SEND THE REQUEST *************
print 'Request:\n\t' + request_parameters

response = requests.post(endpoint, data=request_parameters, headers=headers)
```

```
print 'Response:\n\t' + response.text
```

Amazon Translate에 대한 인증 및 액세스 제어

Amazon Translate에 액세스하려면 AWS가 요청을 인증하는 데 사용할 수 있는 자격 증명이 필요합니다. 이러한 자격 증명은 Amazon Translate 작업에 액세스할 수 있는 권한이 있어야 합니다. 다음 단원에서는 AWS Identity and Access Management(IAM) 및 Amazon Translate를 사용하여 리소스에 액세스할 수 있는 사용자 제어를 통해 리소스를 보호하는 방법에 대한 세부 정보를 제공합니다.

인증

다음과 같은 자격 증명 유형으로 AWS에 액세스할 수 있습니다.

- AWS 계정 루트 사용자 – AWS 계정을 처음 생성할 때는 해당 계정의 모든 AWS 서비스와 리소스에 대해 완전한 액세스 권한이 있는 SSO(single sign-in) 자격 증명으로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 관리 작업이라 할지라도 일상적인 작업에 루트 사용자를 사용하지 마십시오. 대신, IAM 사용자를 처음 생성할 때만 루트 사용자를 사용하는 모범 사례를 준수하십시오. 그런 다음 루트 사용자를 안전하게 보관해 두고 몇 가지 계정 및 서비스 관리 작업을 수행할 때만 자격 증명을 사용합니다.
- IAM 사용자 – IAM 사용자는 특정 사용자 지정 권한(예: Amazon Translate에서 a custom glossary을 만들 권한)이 있는 AWS 계정 내의 자격 증명입니다. IAM 사용자 이름과 암호를 사용하여 AWS Management 콘솔, AWS 토론 포럼 또는 AWS Support Center와 같은 보안 AWS 웹 페이지에 로그인할 수 있습니다.

  사용자 이름과 암호 외에도 각 사용자에 대해 액세스 키를 생성할 수 있습니다. 여러 SDK 중 하나를 통해 또는 AWS Command Line Interface(CLI)를 사용하여 AWS 서비스에 프로그래밍 방식으로 액세스할 때 이러한 키를 사용할 수 있습니다. SDK 및 CLI 도구는 액세스 키를 사용하여 암호화 방식으로 요청에 서명합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. Amazon Translate supports 는 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 지원합니다. 요청 인증에 대한 자세한 내용은 AWS General Reference의 서명 버전 4 서명 프로세스를 참조하십시오.

- IAM 역할 - IAM 역할은 특정 권한을 가진 계정에 생성할 수 있는 IAM 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. IAM 역할을 사용하면 AWS 서비스 및 리소스에 액세스하는 데 사용할 수 있는 임시 액세스 키를 얻을 수 있습니다. 임시 자격 증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

  - 연합된 사용자 액세스 – IAM 사용자를 만드는 대신 AWS Directory Service의 기존 사용자 자격 증명, 엔터프라이즈 사용자 디렉터리 또는 웹 자격 증명 공급자를 사용할 수 있습니다. 이러한 사용자를 연합된 사용자라고 합니다. 자격 증명 공급자를 통해 액세스를 요청하면 AWS가 연합된 사용자에게 역할을 할당합니다. 연합된 사용자에 대한 자세한 내용은 IAM 사용 설명서의 연합된 사용자 및 역할을 참조하십시오.

  - AWS 서비스 액세스 – 계정의 IAM 역할을 사용하여 AWS 서비스에 계정의 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 예를 들어 Amazon Redshift에서 사용자 대신 Amazon S3 버킷에 액세스하

도록 허용하는 역할을 만든 후 그 버킷으로부터 데이터를 Amazon Redshift 클러스터로 로드할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 Creating a Role to Delegate Permissions to an AWS Service 단원을 참조하십시오.

- Amazon EC2에서 실행하는 애플리케이션 – EC2 인스턴스에서 실행되고 AWS API 요청을 하는 애플리케이션의 임시 자격 증명을 IAM 역할을 사용하여 관리할 수 있습니다. 이것은 EC2 인스턴스 내에 액세스 키를 저장하는 경우에 바람직한 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 그 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 만들어야 합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여하기 섹션을 참조하십시오.

액세스 제어

요청을 인증하는 데 유효한 자격 증명이 있더라도, 예를 들어 Amazon Translate 작업을 호출할 수 있는 권한이 있어야 합니다.

다음 단원에서는 Amazon Translate에 대한 권한을 관리하는 방법에 대해 설명합니다. 먼저 개요를 읽어 보면 도움이 됩니다.

- Amazon Translate 리소스에 대한 액세스 권한 관리 개요 (p. 45)
- Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47)

Amazon Translate 리소스에 대한 액세스 권한 관리 개요

작업에 대한 액세스 권한은 권한 정책에서 관리합니다. 계정 관리자는 IAM ID(사용자, 그룹 및 역할)에 권한 정책을 연결하여 작업에 대한 액세스를 관리할 수 있습니다.

Note

계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 정보는 IAM 사용 설명서에서 IAM 모범 사례를 참조하십시오.

권한을 부여할 때 누가 권한을 받는지 그리고 어떤 작업에 대해 권한을 받는지를 결정해야 합니다.

주제

- 작업에 대한 액세스 관리 (p. 45)
- 정책 요소 지정: 리소스, 작업, 효과, 보안 주체 (p. 46)
- 정책에서 조건 지정 (p. 47)

작업에 대한 액세스 관리

권한 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 다음 단원에서는 권한 정책을 만드는 데 사용 가능한 옵션에 대해 설명합니다.

Note

이 단원에서는 Amazon Translate의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. 전체 IAM 설명서는 IAM 사용 설명서의 IAM이란? 단원을

참조하십시오. IAM 정책 구문과 설명에 대한 자세한 정보는 IAM 사용 설명서의 AWS IAM 정책 참조를 참조하십시오.

IAM 자격 증명에 연결된 정책은 자격 증명 기반 정책(IAM 정책)이라고 합니다. 리소스에 연결된 정책은 리소스 기반 정책이라고 합니다. Amazon Translate는 자격 증명 기반 정책만 지원합니다.

자격 증명 기반 정책(IAM 정책)

정책을 IAM 자격 증명에 연결할 수 있습니다. 예를 들면,

- 계정 내 사용자 또는 그룹에 권한 정책 연결 – 사용자 또는 사용자 그룹에 Amazon Translate 작업을 호출할 수 있는 권한을 부여하려면 권한 정책을 특정 사용자 또는 해당 사용자가 속한 그룹에 연결하면 됩니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) – 자격 증명 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어, 계정 A의 관리자는 다음과 같이 다른 AWS 계정(예: 계정 B) 또는 AWS 서비스에 교차 계정 권한을 부여할 역할을 생성할 수 있습니다.
  1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정책을 연결합니다.
  2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
  3. 계정 B 관리자는 계정 B의 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 그러면 계정 B의 사용자가 계정 A에서 리소스를 생성하거나 액세스할 수 있습니다. AWS 서비스에 역할 수임 권한을 부여할 경우 신뢰 정책의 보안 주체가 AWS 서비스 보안 주체일 수도 있습니다.

  IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리를 참조하십시오.

Amazon Translate에서 자격 증명 기반 정책을 사용하는 방법에 대한 자세한 내용은 Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용 (p. 47) 단원을 참조하십시오. 사용자, 그룹, 역할 및 권한에 대한 자세한 내용은 IAM 사용 설명서의 자격 증명(사용자, 그룹 및 역할)을 참조하십시오.

리소스 기반 정책

Lambda과 같은 다른 서비스도 리소스 기반 권한 정책을 지원합니다. 예를 들어, 정책을 S3 버킷에 연결하여 해당 버킷에 대한 액세스 권한을 관리할 수 있습니다. Amazon Translate의 경우 리소스 기반 정책을 지원하지 않습니다.

정책 요소 지정: 리소스, 작업, 효과, 보안 주체

Amazon Translate은 일련의 API 작업을 정의합니다(Actions (p. 59) 참조). 이러한 API 작업에 대한 권한을 부여하기 위해 Amazon Translate에서는 정책에서 지정할 수 있는 작업을 정의합니다.

다음은 가장 기본적인 정책 요소입니다.

- 리소스 – 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. Amazon Translate의 경우 리소스는 항상 `*`입니다.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부할 작업을 식별합니다. 예를 들어 지정한 `Effect`에 따라 `translate:TranslateText`은 Amazon Translate `TranslateText` 작업을 수행할 수 있는 사용자 권한을 허용하거나 거부합니다.
- 결과 – 사용자가 특정 작업을 요청하는 경우의 결과를—허용 또는 거부로 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 리소스에 대한 액세스를 명시적으로 거부할 수도 있습니다. 다른 정책에서 액세스 권한을 부여하더라도 사용자가 해당 리소스에 액세스할 수 없도록 하려고 할 때 이러한 작업을 수행할 수 있습니다.
- 보안 주체 – 자격 증명 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다.

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조를 참조하십시오.

모든 Amazon Translate API 작업을 보여 주는 표는 Amazon Translate API 권한: 작업, 리소스 및 조건 참조 (p. 49) 단원을 참조하십시오.

정책에서 조건 지정

권한을 부여할 때 IAM 정책 언어를 사용하여 정책이 적용되는 조건을 지정합니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건 단원을 참조하십시오.

AWS는 액세스 제어를 위해 IAM을 지원하는 모든 AWS 서비스에 대해 사전 정의된 조건 키 집합을 제공합니다. 예를 들어, 작업을 요청할 때 특정 AWS ID를 요구하려면 `aws:userid` 조건 키를 사용할 수 있습니다. AWS 차원 키에 대한 자세한 내용과 전체 목록은 IAM 사용 설명서의 사용 가능한 조건 키를 참조하십시오.

> Note
>
> 조건 키는 대/소문자를 구분합니다.

Amazon Translate은 조건 키를 추가로 제공하지 않습니다.

Amazon Translate에 대한 자격 증명 기반 정책(IAM 정책) 사용

이 단원에서는 ID 기반 정책의 예를 통해 계정 관리자가 IAM ID(사용자, 그룹, 역할)에 권한 정책을 연결함으로써 Amazon Translate 작업 수행 권한을 부여하는 방법을 보여 줍니다.

> Important
>
> 계속해서 진행하기 전에 Amazon Translate 리소스에 대한 액세스 권한 관리 개요 (p. 45) 단원을 검토해 보는 것이 좋습니다.

다음은 Amazon Translate 및 Amazon Translate 콘솔을 사용하는 데 필요한 권한 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "translate:*",
                "comprehend:DetectDominantLanguage",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
             ],
            "Resource": "*"

        }
    ]
}
```

이 정책에는 네 개의 설명문이 있습니다. 첫째 명령문은 `TranslateText` 작업을 사용할 수 있는 권한을 부여합니다. 둘째 명령문은 언어 자동 감지를 활성화하는 Amazon Comprehend `DetectDominantLanguage` 작업에 대한 권한을 부여합니다. 마지막 두 개의 권한 부여 권한은 Amazon Cloudwatch 서비스를 사용하여 지표에 대한 지원을 제공합니다.

권한을 얻을 보안 주체를 자격 증명 기반 정책에 지정하지 않았으므로 이 정책은 `Principal` 요소를 지정하지 않습니다. 정책을 사용자에게 연결할 경우 사용자는 암시적인 보안 주체입니다. IAM 역할에 권한 정책을 연결하면 역할의 신뢰 정책에서 식별된 보안 주체가 권한을 얻습니다.

KMS 정책은 KMS CMKS를 Amazon Translate 사용자 지정 용어와 함께 사용할 때 필요합니다.

KMS CMKS를 Amazon Translate 사용자 지정 용어와 함께 사용하는 경우 다음 권한 정책 중 하나 이상이 필요할 수 있습니다.

KMS CMK를 사용하여 ImportTerminology를 호출할 수 있으려면 KMS 키 정책에(IAM 정책과는 반대로) 다음 권한이 있어야 합니다.

```
{
    "Id": "key-consolepolicy-3",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow access for use with Amazon Translate",
            "Effect": "Allow",
            "Principal": {
                "AWS": "IAM USER OR ROLE ARN"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:RetireGrant"
            ],
            "Resource": "*"
        }
    ]
}
```

KMS CMK를 사용하여 가져온 사용자 지정 용어에 대해 GetTerminology를 호출할 수 있으려면 KMS 키 정책에 최소한 다음 권한이 있어야 합니다(IAM 정책에는 없음).

```
{
    "Id": "key-consolepolicy-3",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow access for use with Amazon Translate",
            "Effect": "Allow",
            "Principal": {
                "AWS": "IAM USER OR ROLE ARN"
            },
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

KMS CMK를 사용하여 가져온 사용자 지정 용어에 대해 ListTerminologies 또는 DeleteTermionlogy를 호출할 수 있으려면 특수 KMS 권한이 필요 없습니다.

모든 가능성을 처리할 수 있으려면 KMS 키 정책에 최소한 다음 권한이 있어야 합니다(IAM 정책에는 없음).

```
{
    "Id": "key-consolepolicy-3",
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "Allow access for use with Amazon Translate",
            "Effect": "Allow",
            "Principal": {
                "AWS": "IAM USER OR ROLE ARN"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:RetireGrant",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Translate API 작업과 해당 작업이 적용되는 리소스를 담은 표는 Amazon Translate API 권한: 작업, 리소스 및 조건 참조 (p. 49) 단원을 참조하십시오.

Amazon Translate API 권한: 작업, 리소스 및 조건 참조

액세스 제어 (p. 45)을 설정하고 IAM 자격 증명에 연결할 수 있는 권한 정책(자격 증명 기반 정책)을 작성할 때 다음 표를 참조로 사용할 수 있습니다. 이 목록에는 각 Amazon Translate API 작업, 이 작업을 수행할 권한을 부여할 수 있는 작업, 권한을 부여할 수 있는 AWS 리소스가 나와 있습니다. 정책의 `Action` 필드에서 작업을 지정하고, 정책의 `Resource` 필드에서 리소스 값을 지정합니다.

조건을 표현하기 위해서 Amazon Translate 정책에서 AWS 차원 조건 키를 사용하여 조건을 표시할 수 있습니다. AWS 차원 키의 전체 목록은 IAM 사용 설명서의 사용할 수 있는 키 단원을 참조하십시오.

> **Note**
>
> 작업을 지정하려면 `translate:` 접두사 다음에 API 작업 이름을 사용합니다(예: `translate:TranslateText`).

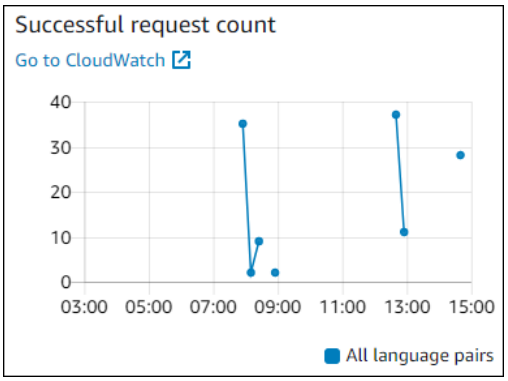
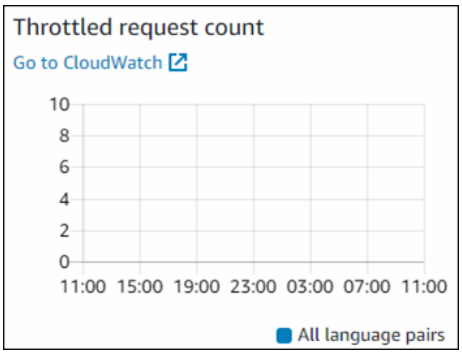
Amazon Translate 모니터링

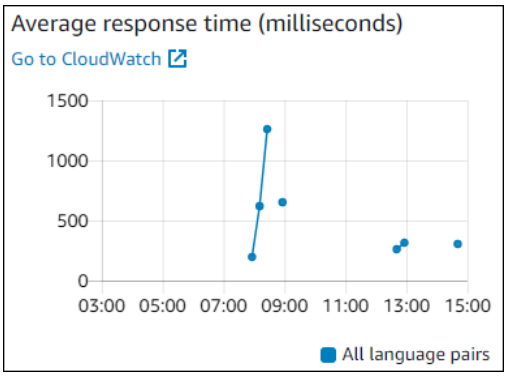
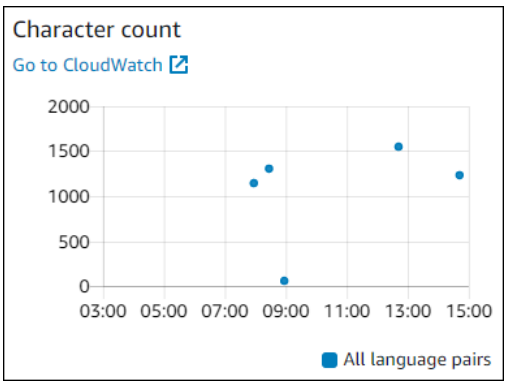
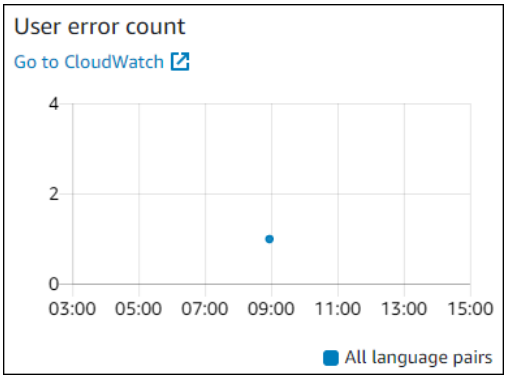
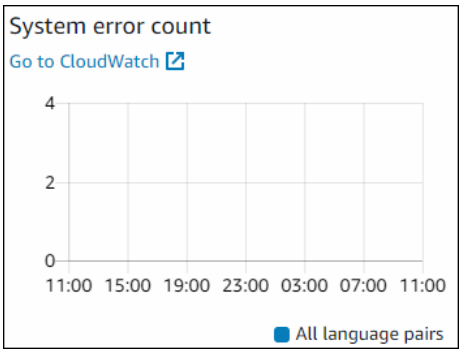
모니터링은 Amazon Translate 및 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어서 중요한 부분입니다. AWS은 Amazon Translate을 모니터링할 수 있는 다양한 도구를 제공합니다. 솔루션을 모니터링하도록 이러한 도구를 구성할 수 있습니다. 모니터링 작업을 최대한 자동화하는 것이 좋습니다.

Amazon Translate는 사전 구성된 그래프를 통해 해당 솔루션에 가장 중요한 측정치를 보여 줍니다. 각 그래프는 솔루션 성능의 특정 측면을 제시합니다. 그래프에 표시되는 시간 범위를 변경하여 시간 변화에 따른 솔루션 성능을 다양한 시각으로 바라볼 수 있습니다.

또한 Amazon CloudWatch를 사용하여 Amazon Translate를 모니터링할 수도 있습니다. CloudWatch로 솔루션의 특정 측정치를 자동으로 모니터링해 보십시오. 어떤 측정치가 설정된 임계값을 벗어날 때마다 알림을 보내 줍니다. CloudWatch API로 필요에 맞는 사용자 지정 모니터링 애플리케이션을 만들 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch란? 단원을 참조하십시오.

다음 표에서는 Amazon Translate의 사전 구성된 그래프를 각각 설명합니다.

그래프	설명
	성공한 요청 수 지정된 기간 동안 Amazon Translate에 보내어 성공한 요청 수입니다.
	스로틀된 요청 수 지정된 기간 동안 Amazon Translate에 보내어 스로틀된 요청 수입니다. 이 정보로 애플리케이션이 Amazon Translate에 요청을 보내는 간격이 너무 빠른지 여부를 알아볼 수 있습니다.

그래프	설명
	평균 응답 시간 지정된 기간 동안 Amazon Translate가 요청을 처리하는 데 걸린 평균 시간입니다.
	문자 수 지정된 기간 동안 Amazon Translate에 보낸 총 문자 수입니다. 이 문자 수에 따라 요금이 청구됩니다.
	사용자 오류 개수 지정된 기간 동안의 사용자 오류 수입니다. 사용자 오류의 HTTP 오류 코드 범위는 400 - 499입니다.
	시스템 오류 개수 지정된 기간 동안의 시스템 오류 수입니다. 시스템 오류의 HTTP 오류 코드 범위는 500 - 599입니다.

Amazon Translate 모니터링

CloudWatch를 사용하면 해당 계정의 개별 Amazon Translate 작업 측정치 또는 전역 Amazon Translate 측정치를 얻을 수 있습니다. 이 측정치를 사용하여 Amazon Translate 기반 솔루션의 상태를 추적하고, 하나 이상의 측정치가 정의된 임계값을 벗어나면 이를 알리도록 경보를 설정할 수 있습니다. 예를 들어 특정 기간 동안 Amazon Translate에 보낸 요청 수를 모니터링하거나, 요청의 지연 시간을 보거나, 오류가 임계값을 초과하면 경보를 발령할 수 있습니다. 지표를 보려면 Amazon Translate 콘솔, Amazon CloudWatch, AWS Command Line Interface 또는 CloudWatch API를 사용할 수 있습니다.

Amazon Translate의 CloudWatch 측정치 이해

Amazon Translate 작업의 측정치를 확인하려면 다음 정보를 지정해야 합니다.

- 측정치 차원. 차원은 측정치를 식별하는 데 사용하는 이름-값 페어 집합입니다. Amazon Translate은 두 가지 차원이 있습니다.
  - `Operation`
  - `Language pair`
- `SuccessfulRequestCount` 또는 `RequestCharacters`와 같은 측정치 이름 전체 측정치 목록은 Amazon Translate의 CloudWatch 측정치 (p. 53) 단원을 참조하십시오.

AWS Management 콘솔, AWS CLI 또는 CloudWatch API를 사용하여 Amazon Translate 측정치를 확인할 수 있습니다. Amazon AWS 소프트웨어 개발 키트(SDK) 또는 CloudWatch API 도구 중 하나를 통해 CloudWatch API를 사용할 수도 있습니다.

아래 표에 CloudWatch 측정치의 몇 가지 일반적인 용도가 나와 있습니다. 모든 사용 사례를 망라한 것은 아니지만 시작하는 데 참고가 될 것입니다.

방법	관련 측정치
성공한 요청 수 추적	`SuccessfulRequestCount` 측정치의 `sum` 통계를 모니터링합니다.
내 애플리케이션이 최대 처리량에 도달했는지 여부 확인	`ThrottledCount` 측정치의 `sum` 통계를 모니터링합니다.
내 애플리케이션의 응답 시간 찾기	`ResponseTime` 측정치의 `average` 통계를 모니터링합니다.
내 애플리케이션의 오류 수 찾기	`ServerErrorCount` 및 `UserErrorCount` 측정치의 `sum` 통계를 모니터링합니다.
요금이 청구되는 문자 수 찾기	`CharacterCount` 측정치의 `sum` 통계를 모니터링합니다.

CloudWatch를 사용하여 Amazon Translate를 모니터링하려면 적절한 CloudWatch 권한이 있어야 합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch에 대한 인증 및 액세스 제어를 참조하십시오.

Amazon Translate 측정치 보기

CloudWatch 콘솔에서 Amazon Translate 측정치 확인

측정치(CloudWatch 콘솔)를 보려면

1. AWS Management 콘솔에 로그인한 다음 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 측정치와 모든 측정치를 선택한 다음 AWS/Translate(AWS/번역)를 선택합니다.
3. 차원과 측정치 이름을 선택한 다음 그래프에 추가를 선택합니다.
4. 날짜 범위 값을 선택합니다. 지정한 날짜 범위의 측정치 개수가 그래프에 표시됩니다.

CloudWatch Amazon Translate의 측정치 및 차원

Amazon Translate의 Amazon CloudWatch 측정치와 차원을 사용하여 솔루션 성능을 모니터링합니다.

Amazon Translate의 CloudWatch 측정치

측정치	설명
`CharacterCount`	요금이 청구되는 요청의 문자 수입니다. 유효한 차원: 언어 쌍, 작업 유효한 통계: 평균, 최대, 최소, 합계 단위: 수
`ResponseTime`	요청에 응답하는 데 걸린 시간입니다. 유효한 차원: 언어 쌍, 작업 유효한 통계: 데이터 샘플, 평균 단위: 데이터 샘플은 개수. 평균 통계는 밀리초입니다.
`ServerErrorCount`	서버 오류 수. 서버 오류의 HTTP 응답 코드 범위는 500 - 599입니다. 유효한 차원: 작업 유효한 통계: 평균, 합계 단위: 수
`SuccessfulRequestCount`	성공한 번역 요청 수입니다. 성공한 요청의 응답 코드는 200 - 299입니다. 유효한 차원: 작업 유효한 통계: 평균, 합계 단위: 수
`ThrottledCount`	스로틀되는 요청 수입니다. `ThrottledCount`로 애플리케이션이 해당 계정에 설정된 요청 수락 속도보다 더 빨리 Amazon Translate에 요청을 보내고 있는지 알아볼 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 Amazon Translate 제한을 참조하십시오. 유효한 차원: 작업 유효한 통계: 평균, 합계

측정치	설명
	단위: 수
`UserErrorCount`	발생한 사용자 오류 수입니다. 사용자 오류의 HTTP 응답 코드 범위는 400 - 499입니다. 유효한 차원: 작업 유효한 통계: 평균, 합계 단위: 수

Amazon Translate에 대한 CloudWatch 차원

다음 차원을 사용하여 Amazon Translate 측정치를 필터링합니다. 소스 언어와 대상 언어로 측정치를 그룹화합니다.

차원	설명
`LanguagePair`	지정한 언어가 포함된 측정치만 사용하도록 제한합니다.
`Operation`	지정한 작업이 있는 측정치만 사용하도록 제한합니다.

지침 및 제한 사항

지원되는 리전

Amazon Translate를 사용할 수 있는 AWS 리전 목록은 Amazon Web Services 일반 참조의 AWS 리전 테이블 또는 AWS 리전 및 엔드포인트를 참조하십시오.

조절

Amazon Translate 조절에 대한 자세한 내용을 알아보고 한도 증가를 요청하려면 Amazon Web Services 일반 참조의 Amazon Translate 제한 사항을 참조하십시오.

- 작업당 조절 속도:

작업	한도
`TranslateText`	언어 쌍별로 20TPS(초당 트랜잭션), 언어 쌍별로 10초당 10kb
`ImportTerminology`	5TPS
`GetTerminology`	10TPS
`ListTerminologies`	10TPS
`DeleteTerminology`	5TPS

지침

Amazon Translate는 분석 모델의 품질을 지속적으로 개선하기 위해 귀하의 콘텐츠를 저장할 수 있습니다. 자세한 내용은 Amazon Translate FAQ를 참조하십시오.

AWS Support에 문의하여 기존 데이터를 삭제하고 계정과 연결된 향후 데이터가 저장되지 않도록 요청할 수 있습니다. 하지만 데이터를 삭제하면 내용에 대한 결과를 개선하는 데 도움이 되는 고유의 교육 데이터가 제거될 수 있기 때문에 Amazon Translate 결과의 품질이 낮아질 수 있습니다.

서비스 제한

Amazon Translate에는 다음과 같은 서비스 제한 사항이 있습니다.

설명	한도
문자 인코딩	UTF-8
문서 크기(UTF-8 문자)	5,000바이트
사용자 지정 용어	

설명	한도
용어 파일 크기	10Mb
용어 파일당 최대 대상 언어 수	10개의 대상 언어
리전별로 AWS 계정당 최대 기존 용어 수	100개의 용어
용어당 최대 소스 텍스트 길이	200바이트
용어당 최대 대상 텍스트 길이	200바이트
TranslateText 응답에 적용된 용어의 최대 용어 수	250개*

* 250개 용어는 소스 텍스트에서 일치된 처음 250개의 용어입니다. 이 제한은 용어당이 아니라, 지정된 TranslateText 요청에서 사용되는 모든 용어에 대한 것입니다. 현재 TranslateText 요청의 용어 수는 1입니다.

Amazon Translate에 대한 문서 이력.

다음 표에서는 본 Amazon Translate 릴리스 관련 문서에 대해 설명합니다.

- 최종 설명서 업데이트: 2019년 5월 8일

update-history-change	update-history-description	update-history-date
새로운 리전	Amazon Translate가 아시아 태평양(몸바이), 아시아 태평양(싱가포르), 아시아 태평양(도쿄) 및 캐나다(중부) 리전에 대한 지원이 추가됩니다. Amazon Translate에서 지원되는 AWS 리전 전체 목록은 지침 및 제한사항 단원을 참조하십시오. 현재 모든 리전에서 모든 언어가 지원되지는 않습니다. 각 리전에서 지원되는 언어 목록은 지원되는 언어단원을 참조하십시오.	May 8, 2019
새로운 언어	Amazon Translate에 힌디어, 말레이어, 노르웨이어, 페르시아어 같은 여러 번역 언어가 추가됩니다. 현재 모든 리전에서 모든 언어가 지원되지는 않습니다. 이러한 예외 사항은 Amazon Translate란? 단원을 참조하십시오. Amazon Translate가 직접 번역할 수 있는 언어 조합은 지원되는 언어 쌍 단원을 참조하십시오.	May 6, 2019
새로운 리전	Amazon Translate에 EU(프랑크푸르트) 및 아시아 태평양(서울) 리전에 대한 지원이 추가됩니다. Amazon Translate에서 지원되는 AWS 리전 전체 목록은 지침 및 제한사항 단원을 참조하십시오.	February 28, 2019
새로운 기능	Amazon Translate에 PCI 규정 준수가 추가됩니다. 자세한 내용은 Amazon Translate란 무엇입니까? 단원을 참조하십시오.	December 12, 2018
새로운 기능	Amazon Translate에 번역을 더 많이 제어할 수 있도록 네 가지 새로운 API와 사용자 지정 용어 기능이 추가됩니다. 사용자 지정 용어를 번역 요청에 사용하면 브랜드 이름, 캐릭터 이름, 모델 이름 및 기타 고유의 내용을 표준 번역이나 해당 문맥과 상관없이 매번 필요한 방식으로 정확하게 번역할 수 있습니다. 자세한 내용은 지원되는 언어 쌍 단원을 참조하십시오.	November 27, 2018

새로운 언어	Amazon Translate는 이제 덴마크어, 네덜란드어, 핀란드어, 히브리어, 인도네시아어, 한국어, 폴란드어 및 스웨덴어로 된 문서를 번역합니다. Amazon Translate는 지원되지 않는 언어 쌍의 수를 크게 줄임으로써 직접 번역을 지속적으로 개선합니다. Amazon Translate가 직접 번역할 수 있는 언어 조합은 지원되는 언어 쌍 단원을 참조하십시오.	November 20, 2018
새로운 기능	Amazon Translate는 영어 이외의 기타 지원되는 언어 간의 직접 번역을 추가합니다. Amazon Translate가 직접 번역할 수 있는 언어 조합은 지원되는 언어 쌍 단원을 참조하십시오.	October 29, 2018
새로운 기능	Amazon Translate에 HIPAA 규정 준수가 추가됩니다. 자세한 내용은 Amazon Translate란 무엇입니까? 단원을 참조하십시오.	October 25, 2018
새로운 기능	Amazon Translate에 중국어(번체), 체코어, 이탈리아어, 일본어, 러시아어, 터키어 같은 여러 가지 새로운 번역 언어가 추가됩니다. Amazon Translate에서 지원하는 언어 목록은 Amazon Translate란 무엇입니까?단원을 참조하십시오.	July 17, 2018
새로운 기능	Amazon Translate는 소스 언어 자동 감지에 대한 지원을 추가합니다. 자세한 내용은 Amazon Translate의 작동 방식 단원을 참조하십시오.	April 4, 2018
새 가이드 (p. 57)	이 문서는 첫 번째 Amazon Translate 개발자 안내서 릴리스입니다.	November 29, 2017

API Reference

이 단원은 API 참조 문서를 포함합니다.

HTTP 헤더

Amazon Translate HTTP 작업에는 또한 일반적인 HTTP 헤더 외에 다음과 같은 필수 헤더가 있습니다.

헤더	값	설명
Content-Type:	application/x-amz-json-1.1	요청 콘텐츠가 JSON이라고 지정합니다. 또한 JSON 버전을 지정합니다.
X-Amz-Date:	<날짜>	Authorization 헤더에 저장되는 서명을 생성할 때 사용할 수 있는 날짜입니다. 형식은 YYYYMMDD'T'HHMMSS'Z' 형식의 ISO 8601 기본이어야 합니다. 예를 들어 다음 날짜/시간 20180820T184626Z는 Amazon Translate에 사용할 수 있는 유효한 x-amz-date입니다. 권한 부여 헤더 사용에 대한 자세한 내용은 Amazon Translate와 함께 서명 버전 4 사용을 참조하십시오.
X-Amz-Target:	AWSShineFrontendService_20170701.<작업>	대상 Amazon Translate 작업. 예를 들어 `TranslateText` 작업을 호출하는 데 `AWSShineFrontendService_20170701.Tra` 사용합니다.

Actions

The following actions are supported:

- DeleteTerminology (p. 60)
- GetTerminology (p. 62)
- ImportTerminology (p. 65)
- ListTerminologies (p. 68)
- TranslateText (p. 71)

DeleteTerminology

A synchronous action that deletes a custom terminology.

Request Syntax

```
{
   "Name": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 84).

The request accepts the following data in JSON format.

Name (p. 60)

The name of the custom terminology being deleted.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^([A-Za-z0-9-]_?)+$`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 82).

InternalServerException

An internal server error occurred. Retry your request.

HTTP Status Code: 500

ResourceNotFoundException

The resource you are looking for has not been found. Review the resource you're looking for and see if a different resource will accomplish your needs before retrying the revised request. .

HTTP Status Code: 400

TooManyRequestsException

You have made too many requests within a short period of time. Wait for a short time and then try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

GetTerminology

Retrieves a custom terminology.

Request Syntax

```
{
   "Name": "string",
   "TerminologyDataFormat": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 84).

The request accepts the following data in JSON format.

Name (p. 62)

The name of the custom terminology being retrieved.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^([A-Za-z0-9-]_?)+$`

Required: Yes

TerminologyDataFormat (p. 62)

The data format of the custom terminology being retrieved, either CSV or TMX.

Type: String

Valid Values: `CSV | TMX`

Required: Yes

Response Syntax

```
{
   "TerminologyDataLocation": {
      "Location": "string",
      "RepositoryType": "string"
   },
   "TerminologyProperties": {
      "Arn": "string",
      "CreatedAt": number,
      "Description": "string",
      "EncryptionKey": {
         "Id": "string",
         "Type": "string"
      },
      "LastUpdatedAt": number,
      "Name": "string",
      "SizeBytes": number,
      "SourceLanguageCode": "string",
      "TargetLanguageCodes": [ "string" ],
```

```
        "TermCount": number
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

TerminologyDataLocation (p. 62)

The data location of the custom terminology being retrieved. The custom terminology file is returned in a presigned url that has a 30 minute expiration.

Type: TerminologyDataLocation (p. 80) object

TerminologyProperties (p. 62)

The properties of the custom terminology being retrieved.

Type: TerminologyProperties (p. 81) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 82).

InternalServerException

An internal server error occurred. Retry your request.

HTTP Status Code: 500

InvalidParameterValueException

The value of the parameter is invalid. Review the value of the parameter you are using to correct it, and then retry your operation.

HTTP Status Code: 400

ResourceNotFoundException

The resource you are looking for has not been found. Review the resource you're looking for and see if a different resource will accomplish your needs before retrying the revised request. .

HTTP Status Code: 400

TooManyRequestsException

You have made too many requests within a short period of time. Wait for a short time and then try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

ImportTerminology

Creates or updates a custom terminology, depending on whether or not one already exists for the given terminology name. Importing a terminology with the same name as an existing one will merge the terminologies based on the chosen merge strategy. Currently, the only supported merge strategy is OVERWRITE, and so the imported terminology will overwrite an existing terminology of the same name.

If you import a terminology that overwrites an existing one, the new terminology take up to 10 minutes to fully propagate and be available for use in a translation due to cache policies with the DataPlane service that performs the translations.

Request Syntax

```
{
   "Description": "string",
   "EncryptionKey": {
      "Id": "string",
      "Type": "string"
   },
   "MergeStrategy": "string",
   "Name": "string",
   "TerminologyData": {
      "File": blob,
      "Format": "string"
   }
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 84).

The request accepts the following data in JSON format.

Description (p. 65)

The description of the custom terminology being imported.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `[\P{M}\p{M}]{0,256}`

Required: No

EncryptionKey (p. 65)

The encryption key for the custom terminology being imported.

Type: EncryptionKey (p. 77) object

Required: No

MergeStrategy (p. 65)

The merge strategy of the custom terminology being imported. Currently, only the OVERWRITE merge strategy is supported. In this case, the imported terminology will overwrite an existing terminology of the same name.

Type: String

Valid Values: `OVERWRITE`

Required: Yes

Name (p. 65)

The name of the custom terminology being imported.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^([A-Za-z0-9-]_?)+$`

Required: Yes

TerminologyData (p. 65)

The terminology data for the custom terminology being imported.

Type: TerminologyData (p. 79) object

Required: Yes

Response Syntax

```
{
   "TerminologyProperties": {
      "Arn": "string",
      "CreatedAt": number,
      "Description": "string",
      "EncryptionKey": {
         "Id": "string",
         "Type": "string"
      },
      "LastUpdatedAt": number,
      "Name": "string",
      "SizeBytes": number,
      "SourceLanguageCode": "string",
      "TargetLanguageCodes": [ "string" ],
      "TermCount": number
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

TerminologyProperties (p. 66)

The properties of the custom terminology being imported.

Type: TerminologyProperties (p. 81) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 82).

InternalServerException

An internal server error occurred. Retry your request.

HTTP Status Code: 500

InvalidParameterValueException

The value of the parameter is invalid. Review the value of the parameter you are using to correct it, and then retry your operation.

HTTP Status Code: 400

LimitExceededException

The specified limit has been exceeded. Review your request and retry it with a quantity below the stated limit.

HTTP Status Code: 400

TooManyRequestsException

You have made too many requests within a short period of time. Wait for a short time and then try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

ListTerminologies

Provides a list of custom terminologies associated with your account.

Request Syntax

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 84).

The request accepts the following data in JSON format.

MaxResults (p. 68)

The maximum number of custom terminologies returned per list request.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 500.

Required: No

NextToken (p. 68)

If the result of the request to ListTerminologies was truncated, include the NextToken to fetch the next group of custom terminologies.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: `\p{ASCII}{0,8192}`

Required: No

Response Syntax

```
{
   "NextToken": "string",
   "TerminologyPropertiesList": [
      {
         "Arn": "string",
         "CreatedAt": number,
         "Description": "string",
         "EncryptionKey": {
            "Id": "string",
            "Type": "string"
         },
         "LastUpdatedAt": number,
         "Name": "string",
         "SizeBytes": number,
         "SourceLanguageCode": "string",
         "TargetLanguageCodes": [ "string" ],
         "TermCount": number
```

```
        }
    ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken (p. 68)

If the response to the ListTerminologies was truncated, the NextToken fetches the next group of custom terminologies.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: `\p{ASCII}{0,8192}`

TerminologyPropertiesList (p. 68)

The properties list of the custom terminologies returned on the list request.

Type: Array of TerminologyProperties (p. 81) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 82).

InternalServerException

An internal server error occurred. Retry your request.

HTTP Status Code: 500

InvalidParameterValueException

The value of the parameter is invalid. Review the value of the parameter you are using to correct it, and then retry your operation.

HTTP Status Code: 400

TooManyRequestsException

You have made too many requests within a short period of time. Wait for a short time and then try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go

- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

TranslateText

Translates input text from the source language to the target language. It is not necessary to use English (en) as either the source or the target language, but not all language combinations are supported by Amazon Translate. For more information, see Supported Language Pairs.

- Arabic (ar)
- Chinese (Simplified) (zh)
- Chinese (Traditional) (zh-TW)
- Czech (cs)
- Danish (da)
- Dutch (nl)
- English (en)
- Finnish (fi)
- French (fr)
- German (de)
- Hebrew (he)
- Hindi (hi)
- Indonesian (id)
- Italian (it)
- Japanese (ja)
- Korean (ko)
- Malay (ms)
- Norwegian (no)
- Persian (fa)
- Polish (pl)
- Portuguese (pt)
- Russian (ru)
- Spanish (es)
- Swedish (sv)
- Turkish (tr)

To have Amazon Translate determine the source language of your text, you can specify `auto` in the `SourceLanguageCode` field. If you specify `auto`, Amazon Translate will call Amazon Comprehend to determine the source language.

Request Syntax

```
{
   "SourceLanguageCode": "string",
   "TargetLanguageCode": "string",
   "TerminologyNames": [ "string" ],
   "Text": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 84).

The request accepts the following data in JSON format.

SourceLanguageCode (p. 71)

The language code for the language of the source text. The language must be a language supported by Amazon Translate.

To have Amazon Translate determine the source language of your text, you can specify `auto` in the `SourceLanguageCode` field. If you specify `auto`, Amazon Translate will call Amazon Comprehend to determine the source language.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 5.

Required: Yes

TargetLanguageCode (p. 71)

The language code requested for the language of the target text. The language must be a language supported by Amazon Translate.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 5.

Required: Yes

TerminologyNames (p. 71)

The TerminologyNames list that is taken as input to the TranslateText request. This has a minimum length of 0 and a maximum length of 1.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^([A-Za-z0-9-]_?)+$`

Required: No

Text (p. 71)

The text to translate. The text string can be a maximum of 5,000 bytes long. Depending on your character set, this may be fewer than 5,000 characters.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 5000.

Pattern: `[\P{M}\p{M}]{1,5000}`

Required: Yes

Response Syntax

```
{
   "AppliedTerminologies": [
      {
         "Name": "string",
         "Terms": [
            {
```

```
                "SourceText": "string",
                "TargetText": "string"
            }
        ]
    }
  ],
  "SourceLanguageCode": "string",
  "TargetLanguageCode": "string",
  "TranslatedText": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AppliedTerminologies (p. 72)

The names of the custom terminologies applied to the input text by Amazon Translate for the translated text response.

Type: Array of AppliedTerminology (p. 76) objects

SourceLanguageCode (p. 72)

The language code for the language of the source text.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 5.

TargetLanguageCode (p. 72)

The language code for the language of the target text.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 5.

TranslatedText (p. 72)

The the translated text. The maximum length of this text is 5kb.

Type: String

Length Constraints: Maximum length of 10000.

Pattern: `[\P{M}\p{M}]{0,10000}`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 82).

DetectedLanguageLowConfidenceException

The confidence that Amazon Comprehend accurately detected the source language is low. If a low confidence level is acceptable for your application, you can use the language in the exception to call Amazon Translate again. For more information, see the DetectDominantLanguage operation in the Amazon Comprehend Developer Guide.

HTTP Status Code: 400

InternalServerException

An internal server error occurred. Retry your request.

HTTP Status Code: 500

InvalidRequestException

The request that you made is invalid. Check your request to determine why it's invalid and then retry the request.

HTTP Status Code: 400

ResourceNotFoundException

The resource you are looking for has not been found. Review the resource you're looking for and see if a different resource will accomplish your needs before retrying the revised request. .

HTTP Status Code: 400

ServiceUnavailableException

The Amazon Translate service is temporarily unavailable. Please wait a bit and then retry your request.

HTTP Status Code: 500

TextSizeLimitExceededException

The size of the text you submitted exceeds the size limit. Reduce the size of the text or use a smaller document and then retry your request.

HTTP Status Code: 400

TooManyRequestsException

You have made too many requests within a short period of time. Wait for a short time and then try your request again.

HTTP Status Code: 400

UnsupportedLanguagePairException

Amazon Translate does not support translation from the language of the source text into the requested target language. For more information, see 예외 처리 (p. 4).

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

Data Types

The following data types are supported:

AppliedTerminology

The custom terminology applied to the input text by Amazon Translate for the translated text response. This is optional in the response and will only be present if you specified terminology input in the request. Currently, only one terminology can be applied per TranslateText request.

Contents

Name

The name of the custom terminology applied to the input text by Amazon Translate for the translated text response.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^([A-Za-z0-9-]_?)+$`

Required: No

Terms

The specific terms of the custom terminology applied to the input text by Amazon Translate for the translated text response. A maximum of 250 terms will be returned, and the specific terms applied will be the first 250 terms in the source text.

Type: Array of Term (p. 78) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for Ruby V2

EncryptionKey

The encryption key used to encrypt the custom terminologies used by Amazon Translate.

Contents

Id

The Amazon Resource Name (ARN) of the encryption key being used to encrypt the custom terminology.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 400.

Pattern: `(arn:aws((-us-gov)|(-iso)|(-iso-b)|(-cn))?:kms:)?([a-z]{2}-[a-z]+(-[a-z]+)?-\d:)?(\d{12}:)?(((key/)?[a-zA-Z0-9-_]+)|(alias/[a-zA-Z0-9:/_-]+))`

Required: Yes

Type

The type of encryption key used by Amazon Translate to encrypt custom terminologies.

Type: String

Valid Values: `KMS`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for Ruby V2

Term

The term being translated by the custom terminology.

Contents

SourceText

The source text of the term being translated by the custom terminology.

Type: String

Length Constraints: Maximum length of 10000.

Pattern: `[\P{M}\p{M}]{0,10000}`

Required: No

TargetText

The target text of the term being translated by the custom terminology.

Type: String

Length Constraints: Maximum length of 10000.

Pattern: `[\P{M}\p{M}]{0,10000}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for Ruby V2

TerminologyData

The data associated with the custom terminology.

Contents

File

The file containing the custom terminology data. Your version of the AWS SDK performs a Base64-encoding on this field before sending a request to the AWS service. Users of the SDK should not perform Base64-encoding themselves.

Type: Base64-encoded binary data object

Length Constraints: Maximum length of 10485760.

Required: Yes

Format

The data format of the custom terminology. Either CSV or TMX.

Type: String

Valid Values: `CSV` | `TMX`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for Ruby V2

TerminologyDataLocation

The location of the custom terminology data.

Contents

Location

The location of the custom terminology data.

Type: String

Length Constraints: Maximum length of 10000.

Pattern: `[\P{M}\p{M}]{0,10000}`

Required: Yes

RepositoryType

The repository type for the custom terminology data.

Type: String

Length Constraints: Maximum length of 10000.

Pattern: `[\P{M}\p{M}]{0,10000}`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for Ruby V2

TerminologyProperties

The properties of the custom terminology.

Contents

Arn

The Amazon Resource Name (ARN) of the custom terminology.

Type: String

Pattern: `^arn:aws((-us-gov)|(-iso)|(-iso-b)|(-cn))?:translate:[a-zA-Z0-9-]+:[0-9]{12}:terminology/.+?/.+?$`

Required: No

CreatedAt

The time at which the custom terminology was created, based on the timestamp.

Type: Timestamp

Required: No

Description

The description of the custom terminology properties.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `[\P{M}\p{M}]{0,256}`

Required: No

EncryptionKey

The encryption key for the custom terminology.

Type: EncryptionKey (p. 77) object

Required: No

LastUpdatedAt

The time at which the custom terminology was last update, based on the timestamp.

Type: Timestamp

Required: No

Name

The name of the custom terminology.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^([A-Za-z0-9-]_?)+$`

Required: No

SizeBytes

The size of the file used when importing a custom terminology.

Type: Integer

Required: No

SourceLanguageCode

The language code for the source text of the translation request for which the custom terminology is being used.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 5.

Required: No

TargetLanguageCodes

The language codes for the target languages available with the custom terminology file. All possible target languages are returned in array.

Type: Array of strings

Length Constraints: Minimum length of 2. Maximum length of 5.

Required: No

TermCount

The number of terms included in the custom terminology.

Type: Integer

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go - Pilot
- AWS SDK for Java
- AWS SDK for Ruby V2

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signature Version 4 Signing Process in the Amazon Web Services General Reference.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: `AWS4-HMAC-SHA256`

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: access_key/YYYYMMDD/region/service/aws4_request.

For more information, see Task 2: Create a String to Sign for Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: `20120325T120000Z`.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to AWS Services That Work with IAM in the IAM User Guide.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Task 1: Create a Canonical Request For Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.