
Amazon Translate

Developer Guide



Amazon Translate: Developer Guide

Copyright © 2019 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon Translate?	1
Are You a First-time User of Amazon Translate ?	1
Supported Language Pairs	2
How It Works	4
Automatic Language Detection	5
Exception Handling	5
Next Steps	5
Getting Started	6
Step 1: Set Up an Account	6
Sign Up for AWS	6
Create an IAM User	7
Next Step	7
Step 2: Set Up the AWS CLI	7
Next Step	8
Step 3: Getting Started (Console)	8
Next Step	9
Step 4: Getting Started (AWS CLI)	9
Translate Text Using the Command Line	10
Translate Text Using a JSON File	10
Next Step	11
Step 5: Getting Started (SDK)	11
Using the SDK for Java	11
Using the AWS SDK for Python	12
Using the Mobile SDK for Android	13
Using the Mobile SDK for iOS	14
Custom Terminology	17
How does this work?	17
Creating a Custom Terminology	17
Compatible Languages	18
Encrypting Your Terminology	19
Best Practices	19
Examples	20
Using Amazon Polly with Amazon Translate	20
Code	20
Using Amazon Translate to Translate a Chat Channel	24
Using Amazon Translate with DynamoDB	31
Example Code	33
Using Amazon Translate to Translate a Web Page	35
Using Amazon Translate to Translate Large Documents	38
Using Signature Version 4 with Amazon Translate	40
Setting Up	40
Code	40
Authentication and Access Control	44
Authentication	44
Access Control	45
Overview of Managing Access	45
Managing Access to Actions	45
Specifying Policy Elements: Resources, Actions, Effects, and Principals	46
Specifying Conditions in a Policy	47
Using Identity-Based Policies (IAM Policies) for Amazon Translate	47
KMS policies needed when using KMS CMKS with Amazon Translate custom terminologies.	48
Amazon Translate API Permissions Reference	49
Monitoring	50
Monitoring with CloudWatch	52

Understanding CloudWatch Metrics for Amazon Translate	52
Viewing Amazon Translate Metrics	53
Creating an Alarm	53
CloudWatch Metrics and Dimensions for Amazon Translate	53
CloudWatch Metrics for Amazon Translate	53
CloudWatch Dimensions for Amazon Translate	54
Guidelines and Limits	56
Supported Regions	56
Throttling	56
Guidelines	56
Service Limits	56
Document History	58
API Reference	60
Actions	60
DeleteTerminology	61
GetTerminology	63
ImportTerminology	66
ListTerminologies	69
TranslateText	72
Data Types	76
AppliedTerminology	77
EncryptionKey	78
Term	79
TerminologyData	80
TerminologyDataLocation	81
TerminologyProperties	82
Common Errors	83
Common Parameters	85
AWS Glossary	88

What Is Amazon Translate?

Amazon Translate uses advanced machine learning technologies to provide high-quality translation on demand. Use it to translate unstructured text documents or to build applications that work in multiple languages.

Amazon Translate translates documents between the following languages:

Arabic	Chinese (Simplified)	Chinese (Traditional)	Czech	Danish
Dutch	English	Finnish	French	German
Hebrew	Indonesian	Italian	Japanese	Korean
Polish	Portuguese	Russian	Spanish	Swedish
Turkish				

Note

Not all translations between language combinations (known as *language pairs*) are currently supported by Amazon Translate. For more information, see [Supported Language Pairs \(p. 2\)](#).

For example, you can:

- Integrate Amazon Translate into your applications to enable multilingual user experiences.
 - Translate company-authored content, such as meeting minutes, technician reports, knowledge-base articles, posts, and more.
 - Translate interpersonal communications, such as email, in-game chat, customer service chat, and more, enabling customers and employees to connect in their preferred language.
- Use Amazon Translate as part of your company's workflow for incoming data.
 - Analyze text, such as social media and news feeds, in many languages.
 - Search information, such as for eDiscovery cases, in many languages.
- Integrate Amazon Translate with other AWS services to enable language-independent processing.
 - Use it with Amazon Comprehend to extract named entities, sentiment, and key phrases from unstructured text, such as social media streams.
 - Use it with Amazon Transcribe to make subtitles and live captioning available in many languages.
 - Use it with Amazon Polly to speak translated content.
 - Use it with Amazon S3 to translate document repositories.
 - Use it with Amazon DynamoDB, Amazon Aurora, and Amazon Redshift to translate text stored in databases.
 - Use it with AWS Lambda or AWS Glue for seamless workflow integration.

This is a HIPAA Eligible Service. For more information about AWS, U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA), and using AWS services to process, store, and transmit protected health information (PHI), see [HIPAA Overview](#).

Are You a First-time User of Amazon Translate ?

If you are a first-time user, we recommend that you read the following sections in order:

1. [How Amazon Translate Works \(p. 4\)](#)—Introduces Amazon Translate.
2. [Getting Started with Amazon Translate \(p. 6\)](#)—Explains how to set up your AWS account and test Amazon Translate.
3. [Examples \(p. 20\)](#)—Provides code examples in Java and Python. Use them to explore how Amazon Translate works.
4. [API Reference \(p. 60\)](#)—Contains reference documentation for Amazon Translate operations.

Supported Language Pairs

Amazon Translate uses advanced machine learning technologies to provide translation between the source language (the input language) to the target language (the output language). The source language-target language combination is known as a *language pair*.

Amazon Translate translates documents between the following languages:

Arabic	Chinese (Simplified)	Chinese (Traditional)	Czech	Danish
Dutch	English	Finnish	French	German
Hebrew	Indonesian	Italian	Japanese	Korean
Polish	Portuguese	Russian	Spanish	Swedish
Turkish				

Not all possible language pairs from the list above are currently supported for translation by Amazon Translate. The following chart shows which languages are supported.

Supported Language Pairs

Note

Rows indicate the source language and columns indicate the target language.

	Ara	Chi (S)	Chi (T)	Cze	Dan	Dut	Eng	Fin	Fre	Ger	Heb	Ind	Itali	Jap	Kor	Poli	Por	Rus	Spa	Swe	Turkish
Arabic	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Chinese (Simpl.)	no	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Chinese (Trad.)	no	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Czech	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Danish	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Dutch	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
English	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Finnish	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
French	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

Amazon Translate Developer Guide
Supported Language Pairs

	Ara	Chi (Sin)	Chi (Tra)	Cze	Dar	Dut	Eng	Fin	Fre	Ger	Heb	Ind	Itali	Jap	Kor	Poli	Por	Rus	Spa	Swe	Turkish
German	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Hebrew	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Indonesian	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes	yes	yes	yes	yes	yes
Italian	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes	yes	yes	yes	yes
Japanese	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes	yes	yes	yes
Korean	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	--	yes	yes	yes	yes	yes	yes
Polish	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes	yes
Portuguese	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes	yes
Russian	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes	yes
Spanish	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes	yes
Swedish	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	yes
Turkish	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	--

How Amazon Translate Works

The Amazon Translate service is based on neural networks trained for language translation. This enables you to translate between a source language (the original language of the text being translated) and a target language (the language into which the text is being translated). It is not necessary to use English 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 \(p. 2\)](#).

When working with Amazon Translate, you will provide source text and get output text:

- **Source text**—The text that you want to translate. You provide the source text in UTF-8 format.
- **Output text**—The text that Amazon Translate has translated into the target language. Output text is also in UTF-8 format. Depending on the source and target languages, there might be more characters in the output text than in the input text.

The translation model has two components, the encoder and the decoder. The *encoder* reads a source sentence one word at a time and constructs a semantic representation that captures its meaning. The *decoder* uses the semantic representation to generate a translation one word at a time in the target language.

Amazon Translate uses attention mechanisms to understand context. This helps it decide which words in the source text are most relevant for generating the next target word. Attention mechanisms enable the decoder to focus on the most relevant parts of a source sentence. This ensures that the decoder correctly translates ambiguous words or phrases.

The target word that the model generates becomes input to the decoder. The network continues generating words until it reaches the end of the sentence.

To translate text, you call the [TranslateText \(p. 72\)](#) method and provide the source text and the target language, using the language code listed in the following table.

Language	Code
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

Language	Code
Indonesian	id
Italian	it
Japanese	ja
Korean	ko
Polish	pl
Portuguese	pt
Russian	ru
Spanish	es
Swedish	sv
Turkish	tr

Automatic Language Detection

Amazon Translate can automatically detect the language used in your source text. To use automatic language detection, specify `auto` as the source language. Amazon Translate calls Amazon Comprehend on your behalf to determine the language used in the source text. By choosing automatic language detection, you agree to the service terms and agreements for Amazon Comprehend. For information about pricing for Amazon Comprehend, see [Amazon Comprehend Pricing](#).

Exception Handling

If you specify a source or target language that isn't supported, Amazon Translate returns the following exceptions:

- **UnsupportedLanguagePairException** – Amazon Translate supports translation between the supported languages. This exception is returned if translation between the language pair is unsupported. For more information, see [Supported Language Pairs \(p. 2\)](#)
- **DetectedLanguageLowConfidenceException** – If you use automatic language detection, and Amazon Translate has low confidence that it detected the correct source language, it returns this exception. If a low confidence level is acceptable, you can use the source language returned in the exception.

Next Steps

Now that you've learned about Amazon Translate you can explore the following sections to learn about creating a solution.

- [Getting Started with Amazon Translate \(p. 6\)](#)
- [Examples \(p. 20\)](#)

Getting Started with Amazon Translate

To get started using Amazon Translate, set up an AWS account and create an AWS Identity and Access Management (IAM) user. To use the AWS Command Line Interface (AWS CLI), download and configure it.

Topics

- [Step 1: Set Up an AWS Account and Create an Administrator User](#) (p. 6)
- [Step 2: Set Up the AWS Command Line Interface \(AWS CLI\)](#) (p. 7)
- [Step 3: Getting Started \(Console\)](#) (p. 8)
- [Step 4: Getting Started \(AWS CLI\)](#) (p. 9)
- [Step 5: Getting Started \(SDK\)](#) (p. 11)

Step 1: Set Up an AWS Account and Create an Administrator User

Before you use Amazon Translate for the first time, complete the following tasks:

1. [Sign Up for AWS](#) (p. 6)
2. [Create an IAM User](#) (p. 7)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all AWS services, including Amazon Translate. You are charged only for the services that you use.

With Amazon Translate, you pay only for the resources that you use. If you are a new AWS customer, you can get started with Amazon Translate for free. For more information, see [AWS Free Usage Tier](#).

If you already have an AWS account, skip to the next section.

To create an AWS account

1. Open <https://aws.amazon.com/>, and then choose **Create an AWS Account**.

Note

If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code using the phone keypad.

Record your AWS account ID because you'll need it for the next task.

Create an IAM User

AWS services, such as Amazon Translate, require that you provide credentials when you access them. This allows the service to determine whether you have permissions to access the service's resources.

We strongly recommend that you access AWS using AWS Identity and Access Management (IAM), not the credentials for your AWS account. To use IAM to access AWS, create an IAM user, add the user to an IAM group with administrative permissions, and then grant administrative permissions to the IAM user. You can then access AWS using a special URL and the IAM user's credentials.

Exercises in this guide assume that you have an IAM user with administrator privileges called `adminuser`.

To create an administrator user

- In your AWS account, create an administrator user called `adminuser`. For instructions, see [Creating Your First IAM User and Administrators Group](#) in the *IAM User Guide*.

For more information about IAM, see the following:

- [AWS Identity and Access Management \(IAM\)](#)
- [Getting Started](#)
- [IAM User Guide](#)

Next Step

[Step 2: Set Up the AWS Command Line Interface \(AWS CLI\) \(p. 7\)](#)

Step 2: Set Up the AWS Command Line Interface (AWS CLI)

You use the AWS CLI to make interactive calls to Amazon Translate.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:
 - [Getting Set Up with the AWS Command Line Interface](#)
 - [Configuring the AWS Command Line Interface](#)
2. In the AWS CLI `config` file, add a named profile for the administrator user:

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

You use this profile when executing AWS CLI commands. For more information about named profiles, see [Named Profiles](#) in the *AWS Command Line Interface User Guide*. For a list of AWS Regions, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

3. Verify the setup by typing the following help command at the command prompt:

```
aws translate help
```

You should see a brief description of Amazon Translate and a list of the available commands.

Next Step

[Step 3: Getting Started \(Console\) \(p. 8\)](#)

Step 3: Getting Started (Console)

The easiest way to get started with Amazon Translate is to use the console to translate some text. You can translate up to 5,000 characters using the console. If you haven't reviewed the concepts and terminology in [How Amazon Translate Works \(p. 4\)](#), we recommend that you do that before proceeding.

To start translating text, sign in to the AWS Management Console and open the Amazon Translate console. If this is the first time that you've used Amazon Translate, choose **Try Amazon Translate**.

In **Translate text**, choose the source and target languages. Enter the text that you want to translate in the left-hand text box. The translated text appears in the right-hand text box.

Try Amazon Translate

Translate text

Source language: English (en) ▼

Target language: German (de)

Amazon Translate uses advanced machine learning technologies to provide high-quality translation on demand. Use it to translate unstructured text documents or to build applications that work in multiple languages.

Amazon Transla... maschinellen L... Übersetzungen... sie, um unstruk...

213 characters, 213 of 5000 bytes used

Is this translation what you expected? Please leave us [feedback](#)

In the **JSON samples** section you can see the JSON input and output to the [TranslateText \(p. 72\)](#) operation.

JSON samples

JSON input and output for your AWS CLI or an AWS SDK

Translation

JSON request

```
{
  "Text": "",
  "SourceLanguageCode": "en",
  "TargetLanguageCode": "de"
}
```

JSON response

```
{
  "TranslatedText": "",
  "SourceLanguageCode": "",
  "TargetLanguageCode": ""
}
```

Next Step

[Step 4: Getting Started \(AWS CLI\) \(p. 9\)](#)

Step 4: Getting Started (AWS CLI)

In the following exercises, you use the AWS command line interface (AWS CLI) to translate text. To complete these exercises, you need to be familiar with the CLI and have a text editor. For more information, see [Step 2: Set Up the AWS Command Line Interface \(AWS CLI\) \(p. 7\)](#).

There are two ways to use the CLI to translate text with Amazon Translate. For short text, you can provide the text that you want to translate as a parameter of the `translate-text` command. For longer text, you can provide the source language, target language, and text in a JSON file.

To use Amazon Translate from the command line, you need to know the endpoint and region for the service. For a list of available endpoints and regions, see [Guidelines and Limits \(p. 56\)](#).

Translate Text Using the Command Line

The following example shows how to use the `translate-text` operation from the command line to translate text. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^). At the command line, type the following.

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

The response is the following JSON:

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

Translate Text Using a JSON File

This example shows how to use the `translate-text` operation to translate a longer text block from a JSON file. You can specify the source and target language on the command line, but in this example, you specify them in the JSON file.

Note

The JSON file is formatted for readability. Reformat the "Text" field to remove line breaks. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

To translate text using a JSON file

1. Copy the following text into a JSON file called `translate.json`:

```
{  
  "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. In the AWS CLI, run the following command:

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

The command outputs a JSON file that contains the following JSON text:

```
{  
  "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"  
}
```

Next Step

To see other ways to use Amazon Translate see [Examples \(p. 20\)](#).

Step 5: Getting Started (SDK)

The following examples demonstrate how to use Amazon Translate [TranslateText \(p. 72\)](#) operation using Java and Python. Use them to learn about the `TranslateText` operation and as building blocks for your own applications.

To run the Java examples, you need to install the AWS SDK for Java. For instructions for installing the SDK for Java, see [Set up the AWS SDK for Java](#).

Topics

- [Translating Text Using the AWS SDK for Java \(p. 11\)](#)
- [Translating Text Using the AWS SDK for Python \(Boto\) \(p. 12\)](#)
- [Translating Text Using the AWS Mobile SDK for Android \(p. 13\)](#)
- [Translating Text Using the AWS Mobile SDK for iOS \(p. 14\)](#)

Translating Text Using the AWS SDK for Java

The following example demonstrates using the [TranslateText \(p. 72\)](#) operation in Java. To run this example, you need the AWS SDK for Java. For instructions for installing the SDK for Java, see [Set up the AWS SDK for Java](#).

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;  
import com.amazonaws.auth.BasicAWSCredentials;  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.translate.AWSTranslate;  
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  
        AWSStaticCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();  
  
        AmazonTranslateClient translate = AmazonTranslateClient.standard()  
            .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());
    }
}
```

You can change the source and target languages as long as the language pair selected is supported by Amazon Translate in the correct combination. For more information, see [Supported Language Pairs \(p. 2\)](#)

Translating Text Using the AWS SDK for Python (Boto)

The following example demonstrates using the [TranslateText \(p. 72\)](#) operation in Python. To run it, you must first install Amazon Translate via the AWS CLI. For instructions, see [the section called "Step 2: Set Up the AWS CLI" \(p. 7\)](#).

```
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'))
```

You can change the source and target languages as long as the language pair selected is supported by Amazon Translate in the correct combination. For more information, see [Supported Language Pairs \(p. 2\)](#)

Another example, this one demonstrating using the custom terminologies operations in Python:

```
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")

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

Translating Text Using the AWS Mobile SDK for Android

You can use Amazon Translate in an Android application to translate text.

To configure the example

1. Set up the AWS Mobile SDK for Android. For instructions, see [Android: Setup Options for the SDK in the AWS Mobile Developer Guide](#)
2. Create an IAM user with the minimum required permissions to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#). After you create the user, download the credentials or record the access key and secret access key.
3. Create a new project with Android Studio.
4. Add the following to the dependencies section of your `build.gradle` file.

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

5. Add the following permissions to the `AndroidManifest.xml` file.

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

6. Copy the source code into your project
7. Change the access key and secret access key to the keys that you recorded in step one.

Code

Use the following code to create the example.

```
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());
            }
        });
    }
}
```

Translating Text Using the AWS Mobile SDK for iOS

You can use Amazon Translate in an iOS application to translate text.

To configure the example

1. Create an IAM user with the minimum required permissions to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#). After you create the user, download the credentials or record the access key and secret access key.
2. Install Xcode version 8.0 or later. You can download the latest version of Xcode from the Apple website, <https://developer.apple.com/xcode/>.
3. Install Cocoapods. In a terminal window, run the following command:

```
sudo gem install cocoapods
```

4. Create a project using Xcode. Then, in a terminal window, navigate to the directory that contains your project's `.xcworkspace` file and run the following command:

```
pod init
```

5. Add the core Mobile SDK for iOS components to your pod file:

```
platform :ios, '9.0'  
target :'app name' do  
  use_frameworks!  
  pod 'AWSTranslate', '~> 2.6.19'  
  # other pods  
end
```

6. Install dependencies by running the following command in a terminal window:

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

7. Running 'pod install' creates a new workspace file. Close your Xcode project and then open it using the `./project_name.xcworkspace` file. From now on you should only use this file to open your Xcode project

Rebuild your app after you open it to resolve APIs from the new libraries called in your code.

8. Add the following import statement to your view controller:

```
import AWSTranslate
```

9. Copy the following code into your Xcode project. Update the access key and secret key to the values that you recorded in step 1.

Code

Use the following code to create the example.

```
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)
```

Custom Terminology

Using custom terminology with your translation requests enables you to make sure that your brand names, character names, model names, and other unique content is translated exactly the way you need it, regardless of its context and the Amazon Translate algorithm's decision.

It's easy to set up a terminology file and attach it to your Amazon Translate account. When you translate text, you simply choose to use the custom terminology as well, and any examples of your source word are translated as you want them.

For example, consider the following: *Amazon Family* is a collection of benefits that offers Amazon Prime members exclusive offers, such as up to 20% off subscriptions to diapers, baby food, and more. In France, it's called *Amazon Famille*. If you translate *Amazon Family* into French using Amazon Translate without any additional context, the result is *Famille Amazon* as the output. While this is an accurate translation, it isn't what the Amazon team in France needs. However, if you add context to this, as in: "Have you ever shopped with Amazon Family?", Amazon Translate determines that the program name does not need to be translated, and the result is "Avez-vous déjà fait des achats avec Amazon Family?". This is a good translation, but still not what the Amazon team is looking for. Custom terminologies can solve issues like this. By adding an entry showing that the term *Amazon Family* should be translated as *Amazon Famille* to their custom terminology, the team can make sure that it is translated this way every time, regardless of context. *Amazon Family* is now translated into *Amazon Famille* and "Have you ever shopped with Amazon Family?" is now translated into "Avez-vous déjà fait des achats avec Amazon Famille?"

Topics

- [How does this work? \(p. 17\)](#)
- [Creating a Custom Terminology \(p. 17\)](#)
- [Encrypting Your Terminology \(p. 19\)](#)
- [Best Practices \(p. 19\)](#)

How does this work?

Generally speaking, when a translation request comes in, Amazon Translate reads the source sentence, creates a semantic representation of the content (in other words, it understands it), and generates a translation into the target language.

When a custom terminology is used as part of the translation request, the engine scans the terminology file before returning the final result. When the engine identifies an exact match between a terminology entry and a string in the source text, it locates the appropriate string in the proposed translation and replaces it with the terminology entry. In the Amazon Family example, it first generates the translation "Avez-vous déjà fait des achats avec Amazon Family?" but stops and replaces *Amazon Family* with *Amazon Famille* before providing the response.

Creating a Custom Terminology

You can use either a CSV file or a TMX file with the source text and the target (translated) term for the terminology file. A single source text is used for each term, but there can be multiple target terms, one for each language, as long as the target and source language can be used.

CSV (comma separated values)

The first column contains the source text, and the other columns contain the target translations. The first row consists of the language codes, with the source language in the first column, and target language translations in the others.

en	fr	de	es	
Amazon	Amazon	Amazon	Amazon	

TMX (Translation Memory eXchange)

A TMX file is an XML-type file commonly used by translation software. Although the form is different than the CSV, the content is similar:

```
<?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>
```

These files are then attached to your Amazon Translate account. When a translation job is run and you choose to use the custom terminology, Amazon Translate then uses the designated word whenever it encounters the source word.

Compatible Languages

Some languages do not change the shape of a word based on sentence context. With these languages, applying a custom terminology is most likely to improve overall translation quality. However, some languages do have extensive word shape changes. We do not recommend applying the feature to those languages, but we do not restrict you from doing so.

The following table shows you languages and recommendations for using this feature:

Language	Recommended/Not Recommended
East Asian Languages (such as Chinese, Japanese, Korean, Indonesian)	Recommended

Language	Recommended/Not Recommended
Germanic Languages (German, Dutch, English, Swedish, Danish, Norwegian)	Recommended
Romance Languages (Italian, French, Spanish, Portuguese)	Recommended
Hebrew	Recommended
Slavic Languages (Russian, Polish, Czech)	Not recommended
Finno-Ugric Languages (Finnish)	Not recommended
Arabic	Not recommended
Turkish	Not recommended

Encrypting Your Terminology

Amazon Translate endeavors to protect all of your data and your custom terminologies are no different. When created, each custom terminology is encrypted so it accessible only by you.

Three encryption options are available:

- Using AWS encryption. This is the default option and is used to safeguard your information if you don't choose another method.
- Using an encryption key associated with your account. If this is chosen, a menu in the console provides you with a choice of associated encryption keys to use.
- Using an encryption key not associated with your account. If you choose this option, a box is displayed in the console for you to enter the Amazon Resource Name (ARN) of the encryption key.

Best Practices

The following are suggested best practices when using custom terminologies

- Keep your custom terminology minimal. Only include words which you want to control and which are completely unambiguous. Only use words that you know you will never want to use an alternate meaning of, and you want it to only ever be translated in a single way. Ideally, limit the list to proper names, like brand names and product names.
- Do not include different translations for the same source phrase (for example, entry #1-EN: Amazon, FR: Amazon; entry #2-EN: Amazon FR: Amazone).
- Some languages do not change the shape of a word based on sentence context. With these languages, applying a custom terminology is most likely to improve overall translation quality. However, some languages do have extensive word shape changes. We do not recommend applying the feature to those languages, but we do not restrict you from doing so. For more information, see [Compatible Languages \(p. 18\)](#).

Examples

The following examples show ways that you can use Amazon Translate.

Topics

- [Using Amazon Polly with Amazon Translate \(p. 20\)](#)
- [Using Amazon Translate to Translate a Chat Channel \(p. 24\)](#)
- [Using Amazon Translate with Amazon DynamoDB \(p. 31\)](#)
- [Using Amazon Translate to Translate a Web Page \(p. 35\)](#)
- [Using Amazon Translate to Translate Large Documents \(p. 38\)](#)
- [Using Signature Version 4 with Amazon Translate \(p. 40\)](#)

Using Amazon Polly with Amazon Translate

To speak translated text, you can use Amazon Polly with Amazon Translate. In this example you'll create a Web page where you can translate text using Amazon Translate and then speak that text using Amazon Polly. The code can be summarized into the following:

- CSS and HTML to create the Web page.
- Initialization code that creates controllers for Amazon Translate and Amazon Polly.
- A function that reads data from the Web page and calls Amazon Translate.
- A function that reads data from the Web page and calls Amazon Polly.
- Utility functions for managing the Web page.

To configure the example

1. Install and Configure the AWS SDK for JavaScript. For instructions for installing the SDK for JavaScript, see [Installing the SDK for JavaScript](#).
2. Copy the code for the example to an HTML file on your Web server.
3. Update the `<script>` tag to the location where you installed the SDK for JavaScript.
4. Change the region and endpoint to the region where you want to run the Amazon Translate and Amazon Polly operations. For a list of supported regions for Amazon Translate, see [Guidelines and Limits \(p. 56\)](#). For a list of supported regions for Amazon Polly, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.
5. Create an IAM user with the minimum required permissions to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#) and [Using Identity-Based Policies \(IAM Policies\) for Amazon Polly](#) in the *Amazon Polly Developer Guide*.
6. Provide the access ID and secret key of the IAM user created in the previous step.

Code

The following is the complete code of the example Web page. You can copy this code into an HTML file to run the example on your own Web server.

```
<!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="de">de</option>
          <option value="es">es</option>
          <option value="fr">fr</option>
          <option value="pt">pt</option>
          <option value="zh">zh</option>
        </select>
      </th>
      <th align="left">
Target Language Code:
        <select id="targetLanguageCodeDropdown">
          <option value="en">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>
      </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();
  </script>

```

```
/**
 * Change the region and endpoint.
 * AWS.config.region = 'region'; // Region
 * var ep = new AWS.Endpoint('endpoint');

/**
 * Place your credentials here. 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({endpoint: ep, 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>
```

Using Amazon Translate to Translate a Chat Channel

You can use Amazon Translate for real time translation of chat messages. This example uses a Twitch channel, but you can use it as a starting point for other real-time streaming text like other chat platforms, customer service interactions, message boards, and more.

This example uses a web page that shows real-time messages in English and their real-time translations side-by-side. You can send the messages to Amazon Polly to speak the text. To follow a person in the chat, type their user name. The app will speak only messages from that user.

The code can be summarized as follows:

- CSS and HTML to create the Web page.
- Initialization code that creates controllers for Amazon Translate and Amazon Polly.
- A call back function that gets executed when a chat message is received.
- A function that sends a chat message.
- A function that calls Amazon Translate to translate messages.
- A function that calls Amazon Polly to synthesize speech.
- Utility functions for managing the Web page.

To configure the example

1. Install and Configure the AWS SDK for JavaScript. For instructions for installing the SDK for JavaScript, see [Installing the SDK for JavaScript](#).
2. Copy the code for the example to an HTML file on your Web server.
3. Update the `<script>` tag to the location where you installed the SDK for JavaScript.
4. Change the region and endpoint to the region where you want to run the Amazon Translate and Amazon Polly operations. For a list of supported regions for Amazon Translate, see [Guidelines and Limits \(p. 56\)](#). For a list of supported regions for Amazon Polly, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.
5. Create an IAM user with the minimum required permissions to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based](#)

[Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#) and [Using Identity-Based Policies \(IAM Policies\) for Amazon Polly](#) in the *Amazon Polly Developer Guide*.

6. Provide the access ID and secret key of the IAM user created in the previous step.
7. Provide a Twitch user name and OAuth token for your account. You can create a Twitch account at <https://www.twitch.tv>. You can create a Twitch OAuth token at <https://twitchapps.com/tmi>.

```
<!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-BVYiisSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+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>
```

```

<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-KJ3o2DKtIkvYIK3UENzmM7KChRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIPg9mGCD8wGNIcPD7Txa"
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-
eEOn7sm1W7DOUI2Xh5I4qSpZTe6hupAO0ovLfqEy0yVJtGRBNfssdmjbJhEYm6Bw"
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

This example shows you how to translate a product review and store it in Amazon DynamoDB. If you request the same review later, DynamoDB returns it without Amazon Translate needing to translate it again.

In this example, you:

- Use AWS CloudFormation to create DynamoDB tables to store the translation and a Lambda function that calls the [TranslateText \(p. 72\)](#) operation.
- Test the function using the AWS Lambda console.

To run the example

1. Copy the contents of `example.py`, which you can find in [Python Lambda Function \(p. 33\)](#), to a file named `example.py`. `example.py` is a Lambda function that calls the [TranslateText \(p. 72\)](#) operation. Compress the file to a zip archive named `example.zip`. Store it in an S3 bucket in the same AWS Region where you want to run the function.
2. Create a new file named `template.yaml`. Copy the AWS CloudFormation template code, which you can find in [AWS CloudFormation Template \(p. 34\)](#), into the file. AWS CloudFormation uses the template to create resources for the sample application. Change `BUCKET_NAME` to the name of the S3 bucket that contains `example.zip`. Save the file in a local directory.
3. Sign in to the AWS Management Console and open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
4. Choose **Create new stack**.
5. Choose **Upload a template to Amazon S3**, and then choose **Choose file**. Choose `template.yaml`, that you created in Step 2, then **Next**.
6. Type a name for the stack, then choose **Next**.
7. On the **Options** page, choose **Next**.
8. Choose **I acknowledge that AWS CloudFormation might create IAM resources and I acknowledge that AWS CloudFormation might create IAM resources with custom names**. For more information, see [Controlling Access with AWS Identity and Access Management](#) in the *AWS CloudFormation User Guide*.
9. Choose **Create Change Set**.
10. After AWS CloudFormation creates the change set, choose **Execute**. Wait until AWS CloudFormation creates the stack.
11. Sign in to the AWS Management Console and open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
12. Choose the new function. Its name starts with `TestTranslate-ReviewTranslate`.
13. On the function detail page, choose **Test**.
14. For **Event name**, type `TestTranslate`. For **Configure test event**, replace the JSON with the following:

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

Choose **Create**.

15. Make sure that **TestTranslate** is selected, then choose **Test**. When the test finishes, you receive the following message:

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. [Learn more](#) about returning r

```
"¡Hola mundo!"
```

Example Code

Use the following code to create the example.

Python Lambda Function

The following is the contents of the Python Lambda function. The Lambda function call the `TranlateText` operation and passes the review, the source language, and the target language to get the translated review. Save this file as `example.py` and then compress it in a .zip archive called `example.zip`. Save the file in an S3 bucket in the same region that you are running the example..

```
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 Template

The following is the template file that you use with AWS CloudFormation to create and configure the Lambda function and the DynamoDB tables. Use this file when you create the AWS CloudFormation stack for the example. Update `BUCKET_NAME` to the name of the S3 bucket that contains the `example.zip` file and then save it to a local directory as `template.yaml`.

```
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
```

Using Amazon Translate to Translate a Web Page

You can use Amazon Translate to translate the contents of a Web page. The following Java program translates a specified Web page from English to Spanish and creates an HTML file that contains the result of the translation. There are two functions in the program:

- A function that reads data from the source Web page, separates it into HTML elements, and then calls the second function to translate the element. At the end of the document, it writes the results to an HTML file.
- A function that calls the Amazon Translate service to translate the contents of an HTML element.

To configure the example

1. Install and configure the AWS SDK for Java. For instructions for installing the SDK for Java, see [Set up the AWS SDK for Java](#).
2. Install the jsoup Java HTML parser. For instructions, see [jsoup](#).
3. Create an IAM user with the minimum required permissions to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#).
4. Set up the credentials needed to run the sample. For instructions, see [Set up AWS Credentials and Region for Development](#) in the *AWS SDK for Java Developer Guide*.
5. Create a new project in your Java IDE and copy the source code.
6. Change the region and endpoint to the region where you want to run the Amazon Translate operation. For a list of supported regions for Amazon Translate, see [Guidelines and Limits \(p. 56\)](#).

```
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.
    }
}
```

```
}
```

Using Amazon Translate to Translate Large Documents

You can split large documents into smaller parts to keep the total document size below the document size limit. For more information about document size limits, see [Service Limits \(p. 56\)](#). The following Java program breaks long text documents into individual sentences and then translates each sentence from the source language to the target language. The program contains two sections:

- The `SentenceSegmenter` class that is responsible for breaking the source string into individual sentences. The sample uses the Java `BreakIterator` class.
- The `main` function that calls the `Translate` operation for each sentence in the source string. The `main` function also handles authentication with Amazon Translate.

To configure the example

1. Install and configure the AWS SDK for Java. For instructions for installing the SDK for Java, see [Set up the AWS SDK for Java](#).
2. Create an IAM user with the minimum required permissions to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#).
3. Set up the credentials needed to run the sample. For instructions, see [Set up AWS Credentials and Region for Development](#) in the *AWS SDK for Java Developer Guide*.
4. Create a new project in your Java IDE and copy the source code.
5. Change the region to the region where you want to run the Amazon Translate operation. For a list of supported regions for Amazon Translate, see [Guidelines and Limits \(p. 56\)](#).
6. Change the source and target languages to the languages to translate between.
7. Run the sample to see the translated text on standard output.

```
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;
    }
}
```

Using Signature Version 4 with Amazon Translate

This example Python program shows how to use Signature Version 4 to add authentication information to Amazon Translate requests. The example makes a POST request, creates a JSON structure that contains the text to be translated in the body (payload) of the request, and passes authentication information in an Authorization header. For more information about using Signature Version 4, see [Signature Version 4 Signing Process](#) in the *Amazon Web Services General Reference*.

Setting Up

To run the example, perform the following steps:

1. Install the AWS Command Line Interface (AWS CLI). The AWS SDK for Python (Boto) is included when you install the AWS CLI. For instructions, see [Step 2: Set Up the AWS Command Line Interface \(AWS CLI\)](#) (p. 7).
2. Create an AWS Identity and Access Management (IAM) user with the minimum required permission policy to run this example. For information about creating an IAM user, see [Creating an IAM User in Your AWS Account](#) in the *AWS Identity and Access Management User Guide*. For the required permissions policies, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate](#) (p. 47). Record the user access key ID and the secret access key.
3. Place the access key ID and secret access key in environment variables named `AWS_ACCESS_KEY` and `AWS_SECRET_ACCESS_KEY`, respectively. As a best practice, we recommend that you don't embed credentials in code.
4. Create a new file on your computer, copy the code for the example (which you can find in the next section), paste it into the file, and save the file with the extension `.py`.
5. In the code, replace `region` with the name of the AWS Region where you want to run the Amazon Translate `TranslateText` operation. For a list of supported Regions, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Code

The following is the complete code of the example Python program.

After creating request values such as the endpoint URL and the body of the request, the code does the following:

1. Create a canonical request to the Amazon Translate `TranslateText` operation.
2. Create the string to that you hash to create the signature.
3. Calculate the signature.
4. Add the signature to the request header.
5. Send the request to the `TranslateText` operation.

To run the example on your computer, copy the code to a Python file.

```
# 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 getSignatureKey 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
```

Authentication and Access Control for Amazon Translate

Access to Amazon Translate requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access Amazon Translate actions. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and Amazon Translate to help secure your resources by controlling who can access them.

- [Authentication \(p. 44\)](#)
- [Access Control \(p. 45\)](#)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
- **IAM user** – An [IAM user](#) is an identity within your AWS account that has specific custom permissions (for example, permissions to create a custom glossary in Amazon Translate). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. Amazon Translate supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the [AWS General Reference](#).

- **IAM role** – An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
 - **Federated user access** – Instead of creating an IAM user, you can use existing user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.

- **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](#) in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests. For example, you must have permissions to call an Amazon Translate action.

The following sections describe how to manage permissions for Amazon Translate. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your Amazon Translate Resources \(p. 45\)](#)
- [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#)

Overview of Managing Access Permissions to Your Amazon Translate Resources

Permissions to access an action are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) to manage access to actions.

Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions and the actions they get permissions for.

Topics

- [Managing Access to Actions \(p. 45\)](#)
- [Specifying Policy Elements: Resources, Actions, Effects, and Principals \(p. 46\)](#)
- [Specifying Conditions in a Policy \(p. 47\)](#)

Managing Access to Actions

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of Amazon Translate. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies). Policies attached to a resource are referred to as *resource-based* policies. Amazon Translate supports only identity-based policies.

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user or a group of users permissions to call an Amazon Translate action, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – To grant cross-account permissions, you can attach an identity-based permissions policy to an IAM role. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
 2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. If you want to grant an AWS service permissions to assume the role, the principal in the trust policy can also be an AWS service principal.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

For more information about using identity-based policies with Amazon Translate, see [Using Identity-Based Policies \(IAM Policies\) for Amazon Translate \(p. 47\)](#). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Lambda, support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon Translate doesn't support resource-based policies.

Specifying Policy Elements: Resources, Actions, Effects, and Principals

Amazon Translate defines a set of API operations (see [Actions \(p. 60\)](#)). To grant permissions for these API operations, Amazon Translate defines a set of actions that you can specify in a policy.

The following are the most basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For Amazon Translate, the resource is always `*`.

- **Action** – You use action keywords to identify operations that you want to allow or deny. For example, depending on the specified `Effect`, `translate:TranslateText` either allows or denies the user permissions to perform the Amazon Translate `TranslateText` operation.
- **Effect** – You specify the effect of the action that occurs when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user cannot access the resource, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal.

To learn more about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the Amazon Translate API actions, see [Amazon Translate API Permissions: Actions, Resources, and Conditions Reference](#) (p. 49).

Specifying Conditions in a Policy

When you grant permissions, you use the IAM policy language to specify the conditions under which a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

AWS provides a set of predefined condition keys for all AWS services that support IAM for access control. For example, you can use the `aws:user-id` condition key to require a specific AWS ID when requesting an action. For more information and a complete list of AWS-wide keys, see [Available Keys for Conditions](#) in the *IAM User Guide*.

Note

Condition keys are case-sensitive.

Amazon Translate does not provide any additional condition keys.

Using Identity-Based Policies (IAM Policies) for Amazon Translate

This topic provides an example of an identity-based policy that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) to grant permissions to perform Amazon Translate actions.

Important

Before you proceed, we recommend that you review [Overview of Managing Access Permissions to Your Amazon Translate Resources](#) (p. 45).

The following is the permissions policy required to use Amazon Translate and the Amazon Translate console:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "translate:*",
        "comprehend:DetectDominantLanguage",

```

```
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
    ],
    "Resource": "*"
}
]
```

The policy has four statements. The first grants permissions to use the `TranslateText` action. The second grants permissions to the Amazon Comprehend `DetectDominantLanguage` operation to enable automatic language detection. The final two permissions grant permissions to use the Amazon Cloudwatch service to provide metrics support.

The policy doesn't specify the `Principal` element because you don't specify the principal who gets the permissions in an identity-based policy. When you attach a policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

KMS policies needed when using KMS CMKS with Amazon Translate custom terminologies.

If you are using KMS CMKS with Amazon Translate custom terminologies, you may need one or more of the following permissions policies.

If you want to be able to call `ImportTerminology` with a KMS CMK, you must have the following permissions in the KMS key policy (as opposed to your IAM policy):

```
{
  "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": "*"
    }
  ]
}
```

If you want to be able to call `GetTerminology` for a custom terminology that was imported with a KMS CMK, you must have at least the following permissions in the KMS key policy (not in your IAM policy):

```
{
  "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": "*"
  }
]
```

If you want to be able to call `ListTerminologies` or `DeleteTerminology` for a custom terminology that was imported with a KMS CMK, you don't need to have any special KMS permissions.

If you want to have all possibilities covered, you should make sure you have at least the following permissions in the KMS key policy (not in your IAM policy):

```
{
  "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": "*"
    }
  ]
}
```

For a table of the Amazon Translate API actions and the resources that they apply to, see [Amazon Translate API Permissions: Actions, Resources, and Conditions Reference](#) (p. 49).

Amazon Translate API Permissions: Actions, Resources, and Conditions Reference

Use the following table as a reference when setting up [Access Control](#) (p. 45) and writing a permissions policy that you can attach to an IAM identity (an identity-based policy). The list includes each Amazon Translate API operation, the corresponding action for which you can grant permissions to perform the action, and the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field, and you specify the resource value in the policy's `Resource` field.

To express conditions, you can use AWS-wide condition keys in your Amazon Translate policies. For a complete list of AWS-wide keys, see [Available Keys](#) in the *IAM User Guide*.

Note

To specify an action, use the `translate:` prefix followed by the API operation name, for example, `translate:TranslateText`.

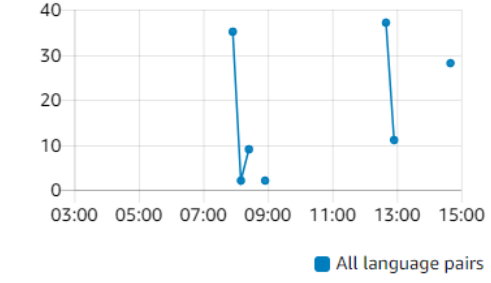
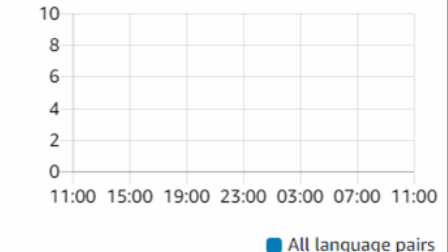
Monitoring Amazon Translate

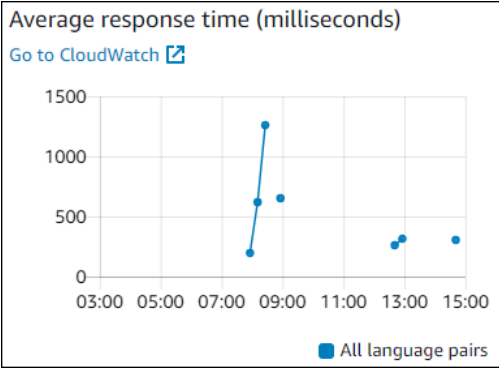
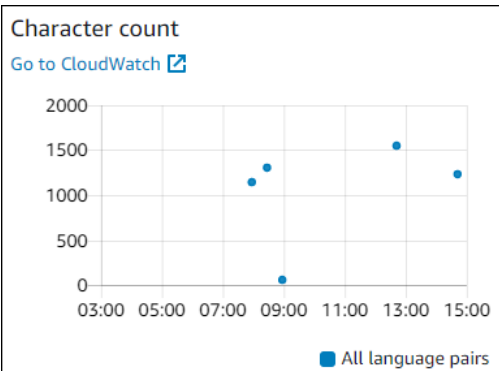
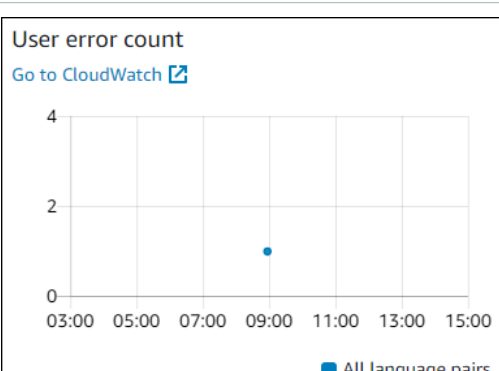
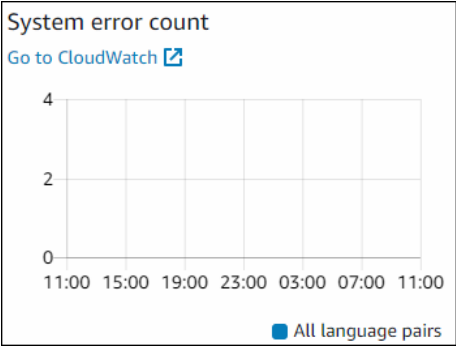
Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Translate and your solutions. AWS provides various tools that you can use to monitor Amazon Translate. You can configure some of these tools to monitor your solutions for you. We recommend that you automate monitoring tasks as much as possible.

Amazon Translate provides preconfigured graphs that show you the most important metrics for your solution. Each graph offers a window into your solution's performance. You can change the time range that the graphs show to get different views of how your solution is performing over time.

You can also use Amazon CloudWatch to monitor Amazon Translate. With CloudWatch, you can automate monitoring specific metrics for your solutions. You receive a notice whenever a metric is outside of the thresholds that you set. You can also use the CloudWatch API to create a custom monitoring application that is suitable for your needs. For more information, see [What is Amazon CloudWatch](#) in the *Amazon CloudWatch User Guide*.

The following table describes each of the preconfigured graphs provided by Amazon Translate.

Graph	Description
<p data-bbox="337 1045 602 1073">Successful request count</p> <p data-bbox="337 1083 516 1108">Go to CloudWatch</p>  <p data-bbox="646 1377 818 1402">■ All language pairs</p>	<p data-bbox="906 1037 1183 1064">Successful request count</p> <p data-bbox="906 1092 1419 1150">The number of successful requests to Amazon Translate during the specified time period.</p>
<p data-bbox="337 1451 591 1478">Throttled request count</p> <p data-bbox="337 1488 516 1514">Go to CloudWatch</p>  <p data-bbox="597 1761 769 1787">■ All language pairs</p>	<p data-bbox="906 1442 1174 1470">Throttled request count</p> <p data-bbox="906 1497 1424 1640">The number of requests to Amazon Translate that were throttled during the specified time period. Use this information to determine if your application is sending request to Amazon Translate too quickly.</p>

Graph	Description														
<p>Average response time (milliseconds) Go to CloudWatch</p>  <table border="1"> <caption>Average response time (milliseconds)</caption> <thead> <tr> <th>Time</th> <th>Average response time (ms)</th> </tr> </thead> <tbody> <tr> <td>08:00</td> <td>~250</td> </tr> <tr> <td>08:30</td> <td>~600</td> </tr> <tr> <td>09:00</td> <td>~1250</td> </tr> <tr> <td>09:30</td> <td>~650</td> </tr> <tr> <td>13:00</td> <td>~300</td> </tr> <tr> <td>14:00</td> <td>~350</td> </tr> </tbody> </table>	Time	Average response time (ms)	08:00	~250	08:30	~600	09:00	~1250	09:30	~650	13:00	~300	14:00	~350	<p>Average response time</p> <p>The average length of time that it took Amazon Translate to process your request in the specified time period.</p>
Time	Average response time (ms)														
08:00	~250														
08:30	~600														
09:00	~1250														
09:30	~650														
13:00	~300														
14:00	~350														
<p>Character count Go to CloudWatch</p>  <table border="1"> <caption>Character count</caption> <thead> <tr> <th>Time</th> <th>Character count</th> </tr> </thead> <tbody> <tr> <td>08:00</td> <td>~1150</td> </tr> <tr> <td>08:30</td> <td>~1300</td> </tr> <tr> <td>09:00</td> <td>~100</td> </tr> <tr> <td>13:00</td> <td>~1550</td> </tr> <tr> <td>14:00</td> <td>~1250</td> </tr> </tbody> </table>	Time	Character count	08:00	~1150	08:30	~1300	09:00	~100	13:00	~1550	14:00	~1250	<p>Character count</p> <p>The total number of characters that you sent to Amazon Translate during the time period. This is the number of characters that you will be billed for.</p>		
Time	Character count														
08:00	~1150														
08:30	~1300														
09:00	~100														
13:00	~1550														
14:00	~1250														
<p>User error count Go to CloudWatch</p>  <table border="1"> <caption>User error count</caption> <thead> <tr> <th>Time</th> <th>User error count</th> </tr> </thead> <tbody> <tr> <td>09:00</td> <td>1</td> </tr> </tbody> </table>	Time	User error count	09:00	1	<p>User error count</p> <p>The number of user errors in the specified time period. User errors are in the HTTP error code range 400 - 499.</p>										
Time	User error count														
09:00	1														
<p>System error count Go to CloudWatch</p>  <table border="1"> <caption>System error count</caption> <thead> <tr> <th>Time</th> <th>System error count</th> </tr> </thead> <tbody> <tr> <td>11:00 - 11:00</td> <td>0</td> </tr> </tbody> </table>	Time	System error count	11:00 - 11:00	0	<p>System error count</p> <p>The number of system errors in the specified time period. System errors are in the HTTP error code range 500 - 599.</p>										
Time	System error count														
11:00 - 11:00	0														

Monitoring Amazon Translate

With CloudWatch, you can get metrics for individual Amazon Translate operations or global Amazon Translate metrics for your account. Use metrics to track the health of your Amazon Translate-based solution and to set up alarms to notify you when one or more metrics fall outside a defined threshold. For example, you can monitor the number of requests made to Amazon Translate in a particular time period, see the latency of requests, or raise an alarm when errors exceed a threshold. To see metrics, you can use the Amazon Translate console, [Amazon CloudWatch](#), the [AWS Command Line Interface](#), or the [CloudWatch API](#).

Understanding CloudWatch Metrics for Amazon Translate

To get metrics for your Amazon Translate operations, you specify the following information:

- The metric dimension. A *dimension* is a set of name-value pairs that you use to identify a metric. Amazon Translate has two dimensions:
 - `Operation`
 - `Language pair`
- The metric name, such as `SuccessfulRequestCount` or `RequestCharacters`. For a complete list of metrics, see [CloudWatch Metrics for Amazon Translate \(p. 53\)](#).

You can get metrics for Amazon Translate with the AWS Management Console, the AWS CLI, or the CloudWatch API. You can use the CloudWatch API through one of the Amazon AWS Software Development Kits (SDKs) or the CloudWatch API tools.

The following table lists some common uses for CloudWatch metrics. These are suggestions to get you started, not a comprehensive list.

How do I?	Relevant metric
Track the number of successful requests	Monitor the <code>sum</code> statistic of the <code>SuccessfulRequestCount</code> metric.
Know if my application has reached its maximum throughput	Monitor the <code>sum</code> statistic of the <code>ThrottledCount</code> metric.
Find the response time for my application	Monitor the <code>average</code> statistic of the <code>ResponseTime</code> metric.
Find the number of errors for my application	Monitor the <code>sum</code> statistic of the <code>ServerErrorCount</code> and <code>UserErrorCount</code> metrics.
Find the number of billable characters	Monitor the <code>sum</code> statistic of the <code>CharacterCount</code> metric.

You must have the appropriate CloudWatch permissions to monitor Amazon Translate with CloudWatch. For more information, see [Authentication and Access Control for Amazon CloudWatch](#) in the *Amazon CloudWatch User Guide*.

Viewing Amazon Translate Metrics

View Amazon Translate metrics in the CloudWatch console.

To view metrics (CloudWatch console)

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Metrics**, choose **All Metrics**, and then choose **AWS/Translate**.
3. Choose the dimension, choose a metric name, and choose **Add to graph**.
4. Choose a value for the date range. The metric count for the specified date range is displayed in the graph.

Creating an Alarm

A CloudWatch alarm watches a single metric over a specified time period, and sends a notification to an Amazon SNS topic or Auto Scaling policy. The action or actions are based on the value of the metric relative to a given threshold over a number of time periods that you specify. CloudWatch can also send you an Amazon Simple Notification Service (Amazon SNS) message when the alarm changes state.

CloudWatch alarms invoke actions only when the state has changed and has persisted for the period that you specify.

To set an alarm

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Alarms**, and choose **Create Alarm**.
3. Choose **AWS/Translate Metrics**, and choose a metric.
4. For **Time Range**, choose a time range to monitor, and choose **Next**.
5. Type a **Name** and **Description**.
6. For **Whenever**, choose \geq , and type a maximum value.
7. If you want CloudWatch to send an email when the alarm state is reached, in the **Actions** section, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose a mailing list or choose **New list** and create a new list.
8. Preview the alarm in the **Alarm Preview** section. If you are satisfied with the alarm, choose **Create Alarm**.

CloudWatch Metrics and Dimensions for Amazon Translate

Use the Amazon CloudWatch metrics and dimensions that are available for Amazon Translate to monitor your solution's performance.

CloudWatch Metrics for Amazon Translate

Metric	Description
CharacterCount	The number of billable characters in requests.

Metric	Description
	<p>Valid dimensions: Language pair, Operation</p> <p>Valid statistics: Average, Maximum, Minimum, Sum</p> <p>Unit: Count</p>
ResponseTime	<p>The time that it took to respond to a request.</p> <p>Valid dimensions: Language pair, Operation</p> <p>Valid statistics: Data samples, Average</p> <p>Unit: For Data samples, count. For Average statistics, milliseconds.</p>
ServerErrorCount	<p>The number of server errors. The HTTP response code range for a server error is 500 to 599.</p> <p>Valid dimension: Operation</p> <p>Valid statistics: Average, Sum</p> <p>Unit: Count</p>
SuccessfulRequestCount	<p>The number of successful translation requests. The response code for a successful request is 200 to 299.</p> <p>Valid dimension: Operation</p> <p>Valid statistics: Average, Sum</p> <p>Unit: Count</p>
ThrottledCount	<p>The number of requests subject to throttling. Use <code>ThrottledCount</code> to determine if your application is sending requests to Amazon Translate faster than your account is configured to accept them. For more information, see Amazon Translate Limits in the <i>Amazon Web Services General Reference</i>.</p> <p>Valid dimension: Operation</p> <p>Valid statistics: Average, Sum</p> <p>Unit: Count</p>
UserErrorCount	<p>The number of user errors that occurred. The HTTP response code range for a user error is 400 to 499.</p> <p>Valid dimension: Operation</p> <p>Valid statistics: Average, Sum</p> <p>Unit: Count</p>

CloudWatch Dimensions for Amazon Translate

Use the following dimensions to filter Amazon Translate metrics. Metrics are grouped by the source language and the target language.

Dimension	Description
LanguagePair	Restricts the metrics to only those that contain the specified languages.
Operation	Restricts the metrics to only those with the specified operation.

Guidelines and Limits

Supported Regions

For a list of AWS Regions where Amazon Translate is available, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Throttling

For information about throttling for Amazon Translate and to request a limit increase, see [Amazon Translate Limits](#) in the *Amazon Web Services General Reference*.

- Throttle rate per account: 100 transactions (requests or operations) per second (tps) with a burst limit of 120 tps.

Concurrent connections per account: 90

- Throttle rate per operation:

Operation	Limit
TranslateText	20 transactions per second (TPS) per language pair, 10kb per 10 seconds per language pair
ImportTerminology	5 TPS
GetTerminology	10 TPS
ListTerminologies	10 TPS
DeleteTerminology	5 TPS

Guidelines

Amazon Translate may store your content to continuously improve the quality of its analysis models. See the [Amazon Translate FAQ](#) to learn more.

You can request the deletion of existing data and that future data associated with your account not be stored by contacting AWS Support. However, deleting your data may degrade your Amazon Translate results because it can remove unique training data that is helpful in improving results for your content.

Service Limits

Amazon Translate has the following service limitations:

Description	Limit
Character encoding	UTF-8

Description	Limit
Document size (UTF-8 characters)	5,000 bytes
Custom terminologies	
Terminology file size	10 Mb
Maximum number of target languages per terminology file	10 target languages
Maximum number of existing terminologies per AWS account per region	100 terminologies
Maximum source text length per term	200 bytes
Maximum target text length per term	200 bytes
Maximum number of terms in the applied terminologies of a TranslateText response	250*

* The 250 terms are the first 250 terms matched in the source text. This limit is not per terminology, but rather for all terminologies used in a given TranslateText request. Currently, the number of terminologies in a TranslateText request is limited to 1.

Document History for Amazon Translate

The following table describes the documentation for this release of Amazon Translate.

- **Latest documentation update:** November 27, 2018

update-history-change	update-history-description	update-history-date
New feature	Amazon Translate adds four new APIs and the new Custom Terminology feature to give you more control over your translation. Using a custom terminology with your translation requests enables you to make sure that your brand names, character names, model names, and other unique content is translated exactly the way you need it, every time, regardless of standard translation or its context. on. see Supported Language Pairs .	November 27, 2018
New languages	Amazon Translate adds multiple new languages for translation: Danish, Dutch, Finnish, Hebrew, Indonesian, Korean, Polish, and Swedish. As part of this process, Amazon Translate continues to improve its direct translation by significantly reducing the number of unsupported language pairs. For the language combinations which can be used for direction translation by Amazon Translate, see Supported Language Pairs .	November 20, 2018
New feature	Amazon Translate adds direct translation between supported languages other than English. For the language combinations which can be used for direction translation by Amazon Translate, see Supported Language Pairs .	October 29, 2018
New feature	Amazon Translate adds HIPAA compliance. For more information, see What Is Amazon Translate .	October 25, 2018

New feature	Amazon Translate adds translation of six additional languages. For a list of languages that Amazon Translate supports, see What Is Amazon Translate .	July 17, 2018
New feature	Amazon Translate adds support for automatic source language detection. For more information, see How Amazon Translate Works .	April 4, 2018
New guide (p. 58)	This is the first release of the <i>Amazon Translate Developer Guide</i> .	November 29, 2017

API Reference

This section contains the API Reference documentation.

Actions

The following actions are supported:

- [DeleteTerminology](#) (p. 61)
- [GetTerminology](#) (p. 63)
- [ImportTerminology](#) (p. 66)
- [ListTerminologies](#) (p. 69)
- [TranslateText](#) (p. 72)

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 \(p. 85\)](#).

The request accepts the following data in JSON format.

Name (p. 61)

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 \(p. 83\)](#).

InternalServerErrorException

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 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 \(p. 85\)](#).

The request accepts the following data in JSON format.

Name (p. 63)

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. 63)

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. 63)

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 \(p. 81\)](#) object

TerminologyProperties (p. 63)

The properties of the custom terminology being retrieved.

Type: [TerminologyProperties \(p. 82\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 83\)](#).

InternalServerErrorException

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 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 \(p. 85\)](#).

The request accepts the following data in JSON format.

Description (p. 66)

The description of the custom terminology being imported.

Type: String

Length Constraints: Maximum length of 256.

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

Required: No

EncryptionKey (p. 66)

The encryption key for the custom terminology being imported.

Type: [EncryptionKey \(p. 78\)](#) object

Required: No

MergeStrategy (p. 66)

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. 66)

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. 66)

The terminology data for the custom terminology being imported.

Type: [TerminologyData \(p. 80\)](#) 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. 67)

The properties of the custom terminology being imported.

Type: [TerminologyProperties \(p. 82\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 83\)](#).

InternalServerErrorException

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 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 \(p. 85\)](#).

The request accepts the following data in JSON format.

MaxResults (p. 69)

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. 69)

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. 69)

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. 69)

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

Type: Array of [TerminologyProperties \(p. 82\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 83\)](#).

InternalServerErrorException

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 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)
- Indonesian (id)
- Italian (it)
- Japanese (ja)
- Korean (ko)
- 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 \(p. 85\)](#).

The request accepts the following data in JSON format.

SourceLanguageCode (p. 72)

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. 72)

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. 72)

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. 72)

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. 73)

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

Type: Array of [AppliedTerminology \(p. 77\)](#) objects

SourceLanguageCode (p. 73)

The language code for the language of the source text.

Type: String

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

TargetLanguageCode (p. 73)

The language code for the language of the target text.

Type: String

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

TranslatedText (p. 73)

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 \(p. 83\)](#).

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

InternalServerErrorException

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: 400

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 [Exception Handling \(p. 5\)](#).

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 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](#) (p. 77)
- [EncryptionKey](#) (p. 78)
- [Term](#) (p. 79)
- [TerminologyData](#) (p. 80)
- [TerminologyDataLocation](#) (p. 81)
- [TerminologyProperties](#) (p. 82)

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 \(p. 79\)](#) 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 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)|(-cn))?:kms:)([a-z]{2}-[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 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 Java](#)
- [AWS SDK for Ruby V2](#)

TerminologyData

The data associated with the custom terminology.

Contents

File

The file containing the custom terminology data.

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 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 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)|(-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 \(p. 78\)](#) 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 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'THHMMSS'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*.