# Table of Contents

What is Amazon Translate? ................................................................................................................... 1
Use cases ............................................................................................................................................... 1
First-time user ..................................................................................................................................... 1
Amazon Translate pricing ..................................................................................................................... 2
Amazon Translate API Reference ......................................................................................................... 2
Supported languages ........................................................................................................................... 3
Supported languages .......................................................... Amazon Translate features ......................... 3
Languages supported by Amazon Translate features ....................................................................... 6
How it works ....................................................................................................................................... 7
Supported formats for the input content ............................................................................................ 7
Customizing your translations ............................................................................................................ 7
Automatic language detection ........................................................................................................... 8
Exception handling ............................................................................................................................ 8
Setting up ........................................................................................................................................... 9
Sign up for an AWS account .............................................................................................................. 9
Create an administrative user ............................................................................................................ 9
Set up the AWS CLI .......................................................................................................................... 10
Grant programmatic access ............................................................................................................... 10
Getting started ................................................................................................................................... 12
Getting started (console) ................................................................................................................ 12
Getting started (AWS CLI) .................................................................................................................. 14
  Translate text using the command line .............................................................................................. 15
  Next step ........................................................................................................................................ 15
Getting started (SDK) ....................................................................................................................... 15
  Using the SDK for Java ................................................................................................................... 15
  Using the AWS SDK for Python ...................................................................................................... 16
  Using the Mobile SDK for Android ................................................................................................ 18
  Using the Mobile SDK for iOS ........................................................................................................ 19
Translation processing modes ............................................................................................................ 22
  Real-time translation ...................................................................................................................... 22
  Real-time translation (console) ...................................................................................................... 22
  Real-time translation (API) ............................................................................................................. 27
Asynchronous batch processing ......................................................................................................... 29
  Region availability .......................................................................................................................... 30
  Prerequisites ................................................................................................................................ 30
  Running a job ................................................................................................................................. 34
  Monitoring and analyzing .............................................................................................................. 38
  Getting results ............................................................................................................................... 39
Customizing your translations ........................................................................................................... 41
  Using do-not-translate tags ............................................................................................................. 41
    Using do-not-translate with the console ....................................................................................... 41
    Using do-not-translate with the API .............................................................................................. 42
  Customizing with custom terminology ......................................................................................... 42
    Creating a custom terminology .................................................................................................... 43
    Using custom terminologies ........................................................................................................ 44
    Encrypting your terminology ....................................................................................................... 45
    Best practices .............................................................................................................................. 45
  Using brevity ................................................................................................................................. 46
    Using the brevity setting ................................................................................................................. 46
    Supported languages ................................................................................................................... 47
  Masking profanity ........................................................................................................................... 47
    Using the profanity setting ........................................................................................................... 48
    Unsupported languages ............................................................................................................... 48
  Setting formality ............................................................................................................................ 48
Guidelines

VPC endpoints (AWS PrivateLink) ................................................. 116
Considerations for Amazon Translate VPC endpoints .................................................. 116
Creating an interface VPC endpoint for Amazon Translate ............................................. 116
Creating a VPC endpoint policy for Amazon Translate .................................................... 116

Supported AWS Regions ............................................................. 118

Compliance .............................................................................. 118
Throttling ............................................................................... 118
Guidelines .............................................................................. 118
Service quotas ........................................................................ 118

Document history .................................................................... 121
API reference ......................................................................... 129
AWS Glossary ....................................................................... 130
What is Amazon Translate?

Amazon Translate is a text translation service that uses advanced machine learning technologies to provide high-quality translation on demand. You can use Amazon Translate to translate unstructured text documents or to build applications that work in multiple languages. See Supported languages and language codes (p. 3) for information about the languages that Amazon Translate supports.

Use cases

Use Amazon Translate to do the following:

**Enable multilingual user experiences in your applications by integrating Amazon Translate:**
- 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, so that customers and employees can connect in their preferred language.

**Process and manage your company’s incoming data:**
- Analyze text, such as social media and news feeds, in many languages.
- Search for information, such as for eDiscovery cases, in many languages.

**Enable language-independent processing by integrating Amazon Translate with other AWS services:**
- Extract named entities, sentiment, and key phrases from unstructured text, such as social media streams with Amazon Comprehend.
- Make subtitles and live captioning available in many languages with Amazon Transcribe.
- Speak translated content with Amazon Polly.
- Translate document repositories stored in Amazon S3.
- Translate text stored in the following databases: Amazon DynamoDB, Amazon Aurora, and Amazon Redshift.
- Seamlessly integrate workflows with AWS Lambda or AWS Glue.

Are you a first-time user of Amazon Translate?

If you are a first-time user of Amazon Translate, we recommend that you start with the following topics:

1. How Amazon Translate works (p. 7) – Introduces Amazon Translate.
2. Getting started with Amazon Translate (p. 12) – Explains how to set up your AWS account and start using Amazon Translate.
3. Examples (p. 61) – Provides code examples in Java and Python. Use the examples to explore how Amazon Translate works.
You can also use the following resources to learn about the Amazon Translate service:

- The AWS Machine Learning Blog includes useful articles about Amazon Translate.
- Amazon Translate Deep Dive Video Series provides introductory videos about Amazon Translate.

Amazon Translate pricing

As with other AWS products, there are no contracts or minimum commitments for using Amazon Translate. For more information about the cost of using Amazon Translate, see Amazon Translate Pricing.

Amazon Translate API Reference

The Amazon Translate API Reference is now a separate document. For more information, see Amazon Translate API Reference.
Supported languages and language codes

Amazon Translate provides translation between a source language (the input language) and a target language (the output language). A source language-target language combination is known as a language pair.

**Note**
Amazon Translate does not charge you for translations if you specify the same language for the source language and the target language. If you set the source language to **auto**, you may be charged for using auto detection. For more information, see [Automatic language detection](p. 8).

**Topics**
- Supported languages (p. 3)
- Languages supported by Amazon Translate features (p. 6)

## Supported languages

Amazon Translate supports text translation between the languages listed in the following table. The language code column uses **ISO 639-1** two-digit language codes. For a country variant of a language, the table follows the **RFC 5646** format of appending a dash followed by an **ISO 3166** 2-digit country code. For example, the language code for the Mexican variant of Spanish is **es-MX**.

<table>
<thead>
<tr>
<th>Language</th>
<th>Language code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans</td>
<td>af</td>
</tr>
<tr>
<td>Albanian</td>
<td>sq</td>
</tr>
<tr>
<td>Amharic</td>
<td>am</td>
</tr>
<tr>
<td>Arabic</td>
<td>ar</td>
</tr>
<tr>
<td>Armenian</td>
<td>hy</td>
</tr>
<tr>
<td>Azerbaijani</td>
<td>az</td>
</tr>
<tr>
<td>Bengali</td>
<td>bn</td>
</tr>
<tr>
<td>Bosnian</td>
<td>bs</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>bg</td>
</tr>
<tr>
<td>Catalan</td>
<td>ca</td>
</tr>
<tr>
<td>Chinese (Simplified)</td>
<td>zh</td>
</tr>
<tr>
<td>Chinese (Traditional)</td>
<td>zh-TW</td>
</tr>
<tr>
<td>Croatian</td>
<td>hr</td>
</tr>
<tr>
<td>Czech</td>
<td>cs</td>
</tr>
<tr>
<td>Language</td>
<td>Language code</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Danish</td>
<td>da</td>
</tr>
<tr>
<td>Dari</td>
<td>fa-AF</td>
</tr>
<tr>
<td>Dutch</td>
<td>nl</td>
</tr>
<tr>
<td>English</td>
<td>en</td>
</tr>
<tr>
<td>Estonian</td>
<td>et</td>
</tr>
<tr>
<td>Farsi (Persian)</td>
<td>fa</td>
</tr>
<tr>
<td>Filipino, Tagalog</td>
<td>tl</td>
</tr>
<tr>
<td>Finnish</td>
<td>fi</td>
</tr>
<tr>
<td>French</td>
<td>fr</td>
</tr>
<tr>
<td>French (Canada)</td>
<td>fr-CA</td>
</tr>
<tr>
<td>Georgian</td>
<td>ka</td>
</tr>
<tr>
<td>German</td>
<td>de</td>
</tr>
<tr>
<td>Greek</td>
<td>el</td>
</tr>
<tr>
<td>Gujarati</td>
<td>gu</td>
</tr>
<tr>
<td>Haitian Creole</td>
<td>ht</td>
</tr>
<tr>
<td>Hausa</td>
<td>ha</td>
</tr>
<tr>
<td>Hebrew</td>
<td>he</td>
</tr>
<tr>
<td>Hindi</td>
<td>hi</td>
</tr>
<tr>
<td>Hungarian</td>
<td>hu</td>
</tr>
<tr>
<td>Icelandic</td>
<td>is</td>
</tr>
<tr>
<td>Indonesian</td>
<td>id</td>
</tr>
<tr>
<td>Irish</td>
<td>ga</td>
</tr>
<tr>
<td>Italian</td>
<td>it</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja</td>
</tr>
<tr>
<td>Kannada</td>
<td>kn</td>
</tr>
<tr>
<td>Kazakh</td>
<td>kk</td>
</tr>
<tr>
<td>Korean</td>
<td>ko</td>
</tr>
<tr>
<td>Latvian</td>
<td>lv</td>
</tr>
<tr>
<td>Lithuanian</td>
<td>lt</td>
</tr>
<tr>
<td>Macedonian</td>
<td>mk</td>
</tr>
<tr>
<td>Malay</td>
<td>ms</td>
</tr>
<tr>
<td>Language</td>
<td>Language code</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Malayalam</td>
<td>ml</td>
</tr>
<tr>
<td>Maltese</td>
<td>mt</td>
</tr>
<tr>
<td>Marathi</td>
<td>mr</td>
</tr>
<tr>
<td>Mongolian</td>
<td>mn</td>
</tr>
<tr>
<td>Norwegian (Bokmål)</td>
<td>no</td>
</tr>
<tr>
<td>Pashto</td>
<td>ps</td>
</tr>
<tr>
<td>Polish</td>
<td>pl</td>
</tr>
<tr>
<td>Portuguese (Brazil)</td>
<td>pt</td>
</tr>
<tr>
<td>Portuguese (Portugal)</td>
<td>pt-PT</td>
</tr>
<tr>
<td>Punjabi</td>
<td>pa</td>
</tr>
<tr>
<td>Romanian</td>
<td>ro</td>
</tr>
<tr>
<td>Russian</td>
<td>ru</td>
</tr>
<tr>
<td>Serbian</td>
<td>sr</td>
</tr>
<tr>
<td>Sinhala</td>
<td>si</td>
</tr>
<tr>
<td>Slovak</td>
<td>sk</td>
</tr>
<tr>
<td>Slovenian</td>
<td>sl</td>
</tr>
<tr>
<td>Somali</td>
<td>so</td>
</tr>
<tr>
<td>Spanish</td>
<td>es</td>
</tr>
<tr>
<td>Spanish (Mexico)</td>
<td>es-MX</td>
</tr>
<tr>
<td>Swahili</td>
<td>sw</td>
</tr>
<tr>
<td>Swedish</td>
<td>sv</td>
</tr>
<tr>
<td>Tamil</td>
<td>ta</td>
</tr>
<tr>
<td>Telugu</td>
<td>te</td>
</tr>
<tr>
<td>Thai</td>
<td>th</td>
</tr>
<tr>
<td>Turkish</td>
<td>tr</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>uk</td>
</tr>
<tr>
<td>Urdu</td>
<td>ur</td>
</tr>
<tr>
<td>Uzbek</td>
<td>uz</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>vi</td>
</tr>
<tr>
<td>Welsh</td>
<td>cy</td>
</tr>
</tbody>
</table>
Languages supported by Amazon Translate features

The following sections describe the languages supported by Amazon Translate features.

- Real-time document translation – Supports translations from English to any supported language, and from any supported language to English. For details about real-time translation, see Real-time translation (p. 22).
- Brevity – For the languages supported by this feature, see Using brevity in Amazon Translate (p. 46).
- Profanity masking – For the languages supported by this feature, see Masking profane words and phrases in Amazon Translate (p. 47).
- Formality – For the languages supported by this feature, see Setting formality in Amazon Translate (p. 48).
How Amazon Translate works

Use the Amazon Translate service to translate content from a source language (the language of the input content) to a target language (the language that you select for the translation output). In a batch job, you can translate files from one or more source languages to one or more target languages. For more information about supported languages, see Supported languages and language codes (p. 3).

Topics
- Supported formats for the input content (p. 7)
- Customizing your translations (p. 7)
- Automatic language detection (p. 8)
- Exception handling (p. 8)

Supported formats for the input content

Amazon Translate, supports the following formats for the input content:

- For real-time translations:
  - Input text – Plain text in UTF-8 format. Amazon Translate provides the output content as UTF-8 text.
  - One input file – A file containing plain text (.txt), HTML (.html), or Word (.docx) content. Amazon Translate provides the output content as a file in the same format as the input file.

- For batch translation jobs:
  - Collection of input files – One or more files that you upload to an Amazon S3 location. Supported file formats include plain text (.txt), HTML (.html), Word (.docx), Excel (.xlsx), PowerPoint (.pptx) and XLIFF 1.2 (.xlf). Amazon Translate provides the output content as files. The file format for each output file matches the input file format.

Customizing your translations

You can use the following features to customize the translations that you produce with Amazon Translate:

- Do-not-translate tags – Uses start and end tags to specify content that you don't want to translate (in HTML content).
- Custom terminology – Defines how you want Amazon Translate to translate specific terms, such as brand names.
- Brevity – Reduces the length of the translation output for most translations (compared to the translation output without brevity). Brevity is supported for real-time text translations.
- Profanity – Masks profane words and phrases in your translation output.
- Formality – Sets the level of language formality in your translation output.
- Parallel data – Adapts the translation output to reflect the style, tone, and word choices in the example translation samples that you provide.

For information, see Customizing your translations.
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 all supported languages. This exception is returned if either the source language or target language is unsupported. For more information, see Supported languages (p. 3).
- **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.
Setting up

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

Setting up tasks
• Sign up for an AWS account (p. 9)
• Create an administrative user (p. 9)
• Install and configure the AWS Command Line Interface (AWS CLI) (p. 10)
• Grant programmatic access (p. 10)

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account
2. Follow the online instructions.
   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.
   When you sign up for an AWS account, an AWS account root user is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

   AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing My Account.

Create an administrative user

After you sign up for an AWS account, create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user
1. Sign in to the AWS Management Console as the account owner by choosing Root user and entering your AWS account email address. On the next page, enter your password.
   For help signing in by using root user, see Signing in as the root user in the AWS Sign-In User Guide.
2. Turn on multi-factor authentication (MFA) for your root user.
   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the IAM User Guide.

Create an administrative user
• For your daily administrative tasks, grant administrative access to an administrative user in AWS IAM Identity Center.
For instructions, see Getting started in the AWS IAM Identity Center User Guide.

**Sign in as the administrative user**

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the AWS Sign-In User Guide.

### Install and configure the AWS Command Line Interface (AWS CLI)

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

**To install and configure the AWS CLI**

1. Install the AWS CLI. For instructions, see the following topic in the AWS Command Line Interface User Guide:
   - Installing or updating the latest version of the AWS Command Line Interface

2. Configure the AWS CLI. For instructions, see the following topic in the AWS Command Line Interface User Guide:
   - Configuring the AWS Command Line Interface

### Grant programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

<table>
<thead>
<tr>
<th>Which user needs programmatic access?</th>
<th>To</th>
<th>By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workforce identity (Users managed in IAM Identity Center)</td>
<td>Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.</td>
<td>Following the instructions for the interface that you want to use.</td>
</tr>
</tbody>
</table>

- For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the AWS Command Line Interface User Guide.
- For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the AWS SDKs and Tools Reference Guide.
<table>
<thead>
<tr>
<th>Which user needs programmatic access?</th>
<th>To</th>
<th>By</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM</td>
<td>Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.</td>
<td>Following the instructions in <a href="https://docs.aws.amazon.com/iam/latest/userguide/using-saml.html">Using temporary credentials with AWS resources</a> in the IAM User Guide.</td>
</tr>
<tr>
<td>IAM</td>
<td>(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.</td>
<td>Following the instructions for the interface that you want to use.</td>
</tr>
<tr>
<td></td>
<td>• For the AWS CLI, see <a href="https://docs.aws.amazon.com/cli/latest/userguide/cli-iam-authenticate.html">Authenticating using IAM user credentials</a> in the AWS Command Line Interface User Guide.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For AWS SDKs and tools, see <a href="https://docs.aws.amazon.com/developerguide/using-iam-policy-keys.html">Authenticate using long-term credentials</a> in the AWS SDKs and Tools Reference Guide.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For AWS APIs, see <a href="https://docs.aws.amazon.com/iam/latest/userguide/using-access-keys.html">Managing access keys for IAM users</a> in the IAM User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
Getting started with Amazon Translate

The easiest way to get started with Amazon Translate is to use the console to translate some text. You can also try out the API operations from the command line. You can also install one of the AWS SDKs to use the Amazon Translate API operations.

Topics

• Getting started (console) (p. 12)
• Getting started (AWS CLI) (p. 14)
• Getting started (SDK) (p. 15)

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 10,000 bytes of text using the console. If you haven't reviewed the concepts and terminology in How Amazon Translate works (p. 7), we recommend that you do so before proceeding.

Open the Amazon Translate console.

If this is the first time that you've used Amazon Translate, choose Launch real-time translation.

In Real-time translation, choose the target language. Amazon Translate autodetects the source language, or you can choose a source language. Enter the text that you want to translate in the left-hand text box. The translated text appears in the right-hand text box.
In the Application integration section you can see the JSON input and output for the TranslateText operation.
Getting started (AWS CLI)

In the following exercise, you use the AWS command line interface (AWS CLI) to translate text. To complete the exercise, you need to be familiar with the CLI and have a text editor. For more information, see Install and configure the AWS Command Line Interface (AWS CLI) (p. 10).

To use Amazon Translate from the command line, you need to run the command from a region that supports the Amazon Translate service. For a list of available endpoints and regions, see Amazon Translate Regions and Endpoints in the AWS General Reference.
Translate text using the command line

The following example shows how to use the TranslateText 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.

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

The response is the following JSON:

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

Next step

To see other ways to use Amazon Translate see Examples (p. 61).

Getting started (SDK)

AWS provides SDKs for various computer languages. The SDK manages many of the API connection details for your client, such as signature calculation, request retry handling, and error handling. For more information, see AWS SDKs.

The following examples demonstrate how to use Amazon Translate TranslateText operation using Java, Python and the mobile SDK. Use the SDKs to learn about the Amazon Translate API and as building blocks for your own applications.

Topics

- Translating text using the AWS SDK for Java (p. 15)
- Translating text using the AWS SDK for Python (Boto) (p. 16)
- Translating text using the AWS Mobile SDK for Android (p. 18)
- Translating text using the AWS Mobile SDK for iOS (p. 19)

Translating text using the AWS SDK for Java

The following example demonstrates using the TranslateText 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.

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

public class App {
    private static final String REGION = "region";
    public static void main(String[] args) {
        // Create credentials using a provider chain. For more information, see
        AWSCredentialsProvider awsCreds = DefaultAWSCredentialsProviderChain.getInstance();

        AmazonTranslate translate = AmazonTranslateClient.builder()  
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds.getCredentials()))  
            .withRegion(REGION)  
            .build();

        TranslateTextRequest request = new TranslateTextRequest()  
            .withText("Hello, world")  
            .withSourceLanguageCode("en")  
            .withTargetLanguageCode("es");

        TranslateTextResult result = translate.translateText(request);
        System.out.println(result.getTranslatedText());
    }
}

For a list of supported languages and language codes, see Supported languages and language codes (p. 3)

Translating text using the AWS SDK for Python (Boto)

The following example demonstrates using the TranslateText operation in Python. To run it, you must first install Amazon Translate via the AWS CLI. For instructions, see the section called “Set up the AWS CLI” (p. 10).

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'))

For a list of supported language codes, see Supported languages and language codes (p. 3)

Custom Terminology

Another example, this one demonstrating using the Custom Terminology operations in Python:

#!/usr/bin/env python
# -*- coding: utf-8 -*-
import boto3
translate = boto3.client(service_name='translate')

# The terminology file 'my-first-terminology.csv' has the following contents:


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

file_data = bytearray(data)

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

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

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

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

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

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

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

file_data = bytearray(data)

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

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

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

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

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. Make sure you have the minimum required permissions to run this example. For the required permissions policies, see Identity-based policies for Amazon Translate (p. 95). 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. Make sure you have the minimum required permissions to run this example. For the required
permissions policies, see Identity-based policies for Amazon Translate (p. 95). 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
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 `.xcodeproj` 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'
   end
   # 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)
Translation processing modes

When translating documents, you can use two different translation processing modes: real-time translation or asynchronous batch processing. The mode you use is based on the size and type of the target documents and affects how you submit the translation job and view its results.

- **Real-time translation** (p. 22) – You make a synchronous request to translate a small amount of text (or a text file) and Amazon Translate responds immediately with the translated text.
- **Asynchronous batch processing** (p. 29) – You put a collection of documents in an Amazon Simple Storage Service (Amazon S3) location and start an asynchronous processing job to translate them. Amazon Translate sends the translated output documents to a specified Amazon S3 location.

Real-time translation

Amazon Translate provides real-time document and text translation operations that immediately return the translations. You can use the console or the API to perform real-time translations.

Topics

- Real-time translation using the console (p. 22)
- Real-time translation using the API (p. 27)

Real-time translation using the console

To use the console for real-time translations, paste input text into the **Source language** text box or provide the input text as a file. Optionally, you can set features such as the desired formality level, profanity masking, brevity, and custom terminology.

You can use auto language detection with real-time translations, but you may incur a charge. For more information, see Automatic language detection (p. 8).

Topics

- Translate text (p. 22)
- Translate a document (p. 23)
- View equivalent API request and response data (p. 25)
- Use translation features (p. 26)

Translate text

Use the Amazon Translate console to translate up to 10,000 bytes of text.

1. Open the Amazon Translate console.
2. In the navigation menu on the left, choose **Real-time translation**.
3. For **Source language**, select the language of the source text, or keep the value as **Auto** for auto detection.
4. For **Target language**, select a language.
5. Enter or paste text into the **Source language** text box. The console displays the translated text in the **Target language** text box.

![Translation](image)

---

**Translate a document**

Real-time document translation supports translations from English to any supported language, and from any supported language to English.

To translate a document using the Amazon Translate console:

1. Open the Amazon Translate console.
2. In the navigation menu on the left, choose **Real-time translation**.
3. In the Translation panel, choose the **Documents** tab.
Translation

Source language

Choose a source language

Choose English for either the source language or the target language.

Upload file

Choose file

Supported file extensions include .html and .txt. The maximum file size is 10MB.

Document type

Plain text (.txt)

Additional settings

Translate and download

Is this translation what you expected? Please leave us feedback
4. For **Source language**, select the language of the source text, or select **Auto** for auto detection.

5. For **Target language**, select a language. If the source language is not English, you must select English for the target language.

6. Under **Upload file**, choose **Choose file** and enter the path to the source file. The maximum file size is 100 KB.

7. For **Document type**, select the format of the translation source file. Document translation supports plain text, HTML, or Word (.docx) input files.

8. Choose **Translate**.

    After the translation task completes, choose **Download translation** to download the translated document to your local hard drive. The format of the translated document (text, HTML, or Word) matches the input document.

**View equivalent API request and response data**

After you use the console to translate the input text or document, you can view the equivalent API request data and response data in JSON format.

1. Below the **Translation** panel, expand the **Application integration** panel.

    The console displays the equivalent translation request data in JSON format.
2. You can copy the JSON request to use in a TranslateText or TranslateDocument API operation.
3. The JSON output in the JSON response panel matches the output that the API generates.

Use translation features

To use translation features with the Amazon Translate console:

1. Open the Amazon Translate console.
2. In the navigation menu on the left, choose Real-time translation.
3. Provide the source language, target language, and the input data (text or document) as described in the previous procedures.
4. Under **Additional settings**, you can choose to customize the output of your translation job with the following settings:

**Custom terminology**

Select a custom terminology file. If the file has an entry for a source term in the input text, Amazon Translate uses the translation from the terminology file.

For more information, see [Customizing your translations with custom terminology](p. 42).

**Brevity**

Reduces the length of the translation output for most translations (compared to the translation output without brevity). Amazon Translate supports brevity for translating text, but not for translating a document. Amazon Translate ignores the brevity setting if the source and target language form an unsupported language pair for brevity.

For information about supported languages, see [Using brevity in Amazon Translate](p. 46).

**Profanity**

Masks profane words and phrases in your translation output. Amazon Translate doesn't support profanity masking in all supported languages.

For more information, see [Masking profane words and phrases in Amazon Translate](p. 47).

**Formality**

For some target languages, you can set **Formality** to formal or informal. Amazon Translate ignores the formality setting if formality doesn't support the target language.

For more information, see [Setting formality in Amazon Translate](p. 48).

5. For document translation, choose **Translate** to translate the document using the chosen features.

For text translation, the console applies the translation feature to the translated text when you choose each feature.

---

**Real-time translation using the API**

Amazon Translate provides the following real-time translation operations to support interactive applications:

- **TranslateText** – translates a block of text.
- **TranslateDocument** – translates the contents of a file (plain text, HTML, or .docx).

These synchronous operations return the translation result directly to your application. If you use auto language detection with these operations, you may incur a charge. For more information, see [Automatic language detection](p. 8).

**Translate text**

Use the **TranslateText** operation to translate a single block of text.

**Translate text using the command line**

The following example shows how to use the **TranslateText** operation from the command line. 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, enter the following command.

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

The command responds with the following JSON:

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

**Translate text using a JSON file**

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

**Note**
The example 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`:

   ```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:

   ```json
   {
     "TargetLanguageCode": "fr",
     "TranslatedText": "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"
   }
   ```
Translate document

Use the **TranslateDocument** operation to translate a text, HTML, or Word (.docx) document and return the translation result directly to your application.

Real-time document translation supports translations from English to any supported language, and from any supported language to English. You can specify the source language code or use auto detect.

**Translate document using the command line**

The following example shows how to use the **TranslateDocument** operation from the command line. 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, enter the following command.

```
aws translate translate-document \
   --region region \
   --source-language-code "en" \
   --target-language-code "es" \
   --document-content fileb://source-lang.txt \
   --document Content-Type=text/plain \
   --query "TranslatedDocument.Content" \
   --output text | base64 \
   --decode > target-lang.txt
```

The command responds with the following JSON:

```
{
   "SourceLanguageCode": "en",
   "TargetLanguageCode": "es",
   "TranslatedDocument":{
      "Content": blob
   }
}
```

Asynchronous batch processing with Amazon Translate

To translate large collections of documents (up to 5 GB in size), use the Amazon Translate asynchronous batch processing operation, **StartTextTranslationJob**. This is best for collections of short documents, such as social media postings or user reviews, or any situation in which instantaneous translation is not required.

To perform an asynchronous batch translation, you typically perform the following steps:

1. Store a set of documents in an input folder inside of an Amazon S3 bucket.
2. Start a batch translation job.
3. As part of your request, provide Amazon Translate with an IAM role that has read access to the input Amazon S3 folder and all its sub-folders. The role must also have read and write access to an output Amazon S3 bucket.
4. Monitor the progress of the batch translation job.
5. Retrieve the results of the batch translation job from the specified output bucket.
Region availability

Batch translation is supported in the following AWS Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)

Topics

- Prerequisites for batch translation jobs (p. 30)
- Running a batch translation job (p. 34)
- Monitoring and analyzing batch translation jobs (p. 38)
- Getting batch translation results (p. 39)

Prerequisites for batch translation jobs

The following prerequisites must be met in order for Amazon Translate to perform a successful batch translation job:

- The Amazon S3 buckets that contain your input and output documents must be in the same AWS Region as the API endpoint you are calling.
- The collection of batch input documents must be 5 GB or less in size.
- There can be a maximum of one million documents submitted in a batch translation job.
- Each input document must be 20 MB or less and must contain fewer than 1 million characters.
- Your input files must be in a folder in an Amazon S3 bucket. If you add your input files to the top level of a bucket, Amazon Translate throws an error when you attempt to run a batch translation job. This requirement applies to the input files. No folder is necessary for the output files, and Amazon Translate can place them at the top level of an Amazon S3 bucket.
- Your input file folder can contain nested folders. Make sure none of the nested folders are named details, otherwise Amazon Translate throws an error when you attempt to run the batch translation job.

Supported file formats

Amazon Translate supports the following types of files for batch translation jobs:
Amazon Translate Developer Guide
Prerequisites

- Plain text.
- HTML.
- Word documents (.docx).
- PowerPoint Presentation files (.pptx).
- Excel Workbook files (.xlsx).
- XML Localization Interchange File Format (XLIFF) files (.xlf). Amazon Translate supports only XLIFF version 1.2.

Amazon Translate requires files to be UTF-8 encoded.

**Prerequisite permissions**

Before you can run a batch translation job, your AWS account must have a service role in IAM. This role must have a permissions policy that grants Amazon Translate:

- Read access to your input folder and all its sub-folders in Amazon S3.
- Read and write access to your output bucket.

It must also include a trust policy that allows Amazon Translate to assume the role and gain its permissions. This trust policy must allow the `translate.amazonaws.com` service principal to perform the `sts:AssumeRole` action.

When you create a batch translation job by using the Amazon Translate console, you have the option to allow Amazon Translate to automatically create this role for you. When you run a batch translation job by using the AWS CLI or the Amazon Translate API, you provide the Amazon Resource Name (ARN) of the role in your request.

For more information, see [Creating a Role to Delegate Permissions to an AWS Service](https://docs.aws.amazon.com/IdentityAndAccessManagement/latest/UserGuide/CreateRole.html) in the *IAM User Guide*.

**Example Permissions policy**

The following example permissions policy grants read access to an input folder in an Amazon S3 bucket. It grants read and write access to an output bucket.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "s3:GetObject",
         "Resource": [
            "arn:aws:s3:::input-bucket-name/*",
            "arn:aws:s3:::output-bucket-name/*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": "s3:ListBucket",
         "Resource": [
            "arn:aws:s3:::input-bucket-name",
            "arn:aws:s3:::output-bucket-name"
         ]
      },
      {
         "Effect": "Allow",
         "Action": ["s3:PutObject"
```
Example Trust policy

The following trust policy allows Amazon Translate to assume the IAM role that the policy belongs to.

We recommend that you verify the AWS account that is using the trust policy, to mitigate the **Confused deputy** problem. This example uses the `aws:SourceArn` and `aws:SourceAccount` condition keys to verify the source account. Enter the AWS account that submits the batch translation job.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "translate.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:translate::*:111122223333:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

Prerequisite permissions to customize encryption

You can customize your encryption settings in Amazon Translate, but first you must add permissions to the service role in IAM.

Amazon Translate encrypts the translation output that you produce when you run a batch translation job. By default, it does this encryption with an **AWS managed key**. This type of key is created by AWS and stored in AWS Key Management Service (AWS KMS) in your account. However, you cannot manage this KMS key yourself. It is managed and used on your behalf only by AWS.

Optionally, you can choose to encrypt your output with a **customer managed key**, which is a KMS key that you create, own, and manage in your AWS account.

Your key must have a key policy that enables Amazon Translate to use it. The key policy does this by granting its permissions to the service role that allows Amazon Translate to access your Amazon S3 bucket.

The key policy allows the service role to perform the AWS KMS operations that are required to encrypt your output, as shown by the following example policy statement.

**Example KMS key policy statement**

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "translate.amazonaws.com"
  },
  "Action": "kms:Encrypt",
  "Condition": {
    "ArnLike": {
      "aws:PrincipalArn": "arn:aws:iam::*:111122223333:root"
    }
  }
}
```
Prerequisites

"AWS": [
    "arn:aws:iam::111122223333:role/AmazonTranslateServiceRoleS3FullAccess"
],
"Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey",
    "kms:CreateGrant",
    "kms:RetireGrant",
    "kms:DescribeKey"
],
"Resource": "*"
}

For more information, see [Key policies in AWS KMS](#) in the [AWS Key Management Service Developer Guide](#)

Permissions to use an AWS KMS key from another AWS account

If you want to use a KMS key that's in a different AWS account from the one where you use Amazon Translate, then you must:

1. Update the service role for Amazon Translate in IAM.
2. Update the key policy in AWS KMS.

To update your service role, attach a policy that allows it to perform the necessary AWS KMS operations with the KMS key that's in the other AWS account, as shown by the following example.

**Example IAM policy to grant access to a KMS key in a different account**

```json
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:CreateGrant",
        "kms:RetireGrant",
        "kms:DescribeKey"
    ],
    "Resource": "arn:aws:kms:us-west-2:111122223333:key/key-id"
}
```

To update your KMS key policy, add the service role and admin user as principals that are allowed to use the key, as shown by the following example policy statement.

**Example KMS key policy statement to allow an IAM role to use the key**

```json
{
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/AmazonTranslateServiceRoleS3FullAccess",
            "arn:aws:iam::444455556666:admin"
        ]
    },
    "Action": [
```

```json
33
```
Running a batch translation job

You can run a batch translation job by using the Amazon Translate console, the AWS CLI, or the Amazon Translate API.

**Note**
Batch translation jobs are long-running operations and can take significant time to complete. For example, batch translation on a small dataset might take a few minutes, while very large datasets may take up to 2 days or more. Completion time is also dependent on the availability of resources.

### Amazon Translate console

To run a translation job by using the Amazon Translate console, use the **Batch translation** page to create the job:

1. Open the Amazon Translate console.
2. In the navigation menu on the left, choose **Batch translation**.
3. On the **Translation jobs** page, choose **Create job**. The console shows the **Create translation job** page.
4. **Under Job settings**, do the following:
   a. For **Name**, enter a custom name for the batch translation job.
   b. For **Source language**, select the language of the source files. If you don't know the language of the source files, or your input documents contains different source languages, select **auto**. Amazon Translate auto detects the source language for each file.
   c. For **Target languages**, select up to 10 languages. Amazon Translate translates each source file into each target language.
5. **Under Input data**, do the following:
   a. For **Input S3 location**, specify the input folder that contains the translation source files in Amazon S3. To provide the folder by navigating to it in Amazon S3, choose **Select folder**.
   b. For **File format**, select format of the translation source files.
6. **Under Output data**, do the following:
   a. For **Output S3 location**, specify the output folder in Amazon S3 where Amazon Translate puts the translation output. To provide the folder by navigating to it in Amazon S3, choose **Select folder**.
   b. Optionally, choose **Customize encryption settings (advanced)** if you want to encrypt your output with a customer managed key that you manage in the AWS Key Management Service (AWS KMS).

   By default, Amazon Translate encrypts your translation output using a KMS key that is created, managed, and used on your behalf by AWS. Choose this option if you want to encrypt your output with your own KMS key instead.
If you want to use a KMS key from the current AWS account, select it under **Choose an AWS Key Management Service key**. Or, if you want to use a KMS key from a different AWS account, enter the Amazon Resource Name (ARN) for that key.

**Note**
Before you can use your own KMS key, you must add permissions to the service role for Amazon Translate in IAM. If you want to use a KMS key from a different account, you must also update the key policy in AWS KMS. For more information, see [Prerequisite permissions to customize encryption](#).

7. Under **Customizations - optional**, you can choose to customize the output of your translation job with the following settings:

**Profanity**

Masks profane words and phrases in your translation output. If you specify multiple target languages for the job, all the target languages must support profanity masking. If any of the target languages don't support profanity masking, the translation job won't mask profanity for any target language.

For more information, see [Masking profane words and phrases in Amazon Translate](#).

**Brevity**

Amazon Translate doesn't support brevity for batch translation jobs.

For more information, see [Using brevity in Amazon Translate](#).

**Formality**

For some target languages, you can set **Formality** to formal or informal. If you specify multiple target languages for the job, translate ignores the formality setting for any unsupported target language.

For more information, see [Setting formality in Amazon Translate](#).

**Custom terminology**

Consists of example source terms and the desired translation for each term. If you specify multiple target languages for the job, translate uses the designated terminology for each requested target language that has an entry for the source term in the terminology file.

For more information, see [Customizing your translations with custom terminology](#).

**Parallel data**

Consists of examples that show how you want segments of text to be translated. If you specify multiple target languages for the job, the parallel data file must include translations for all the target languages.

When you add parallel data to a batch translation job, you create an **Active Custom Translation** job.

**Note**
Active Custom Translation jobs are priced at a higher rate than other jobs that don't use parallel data. For more information, see [Amazon Translate pricing](#).

For more information, see [Customizing your translations with parallel data (Active Custom Translation)](#).

8. Under **Access permissions**, provide Amazon Translate with an IAM role that grants the required permissions to your input and output files in Amazon S3:
If you already have this IAM role in your account, choose **Use an existing IAM role**, and select it under **IAM role**.

If you don't already have this IAM role in your account, choose **Create an IAM role**. For **IAM role**, choose **Input and output S3 buckets**. For **Role name**, provide a custom name. When you create the translation job, Amazon Translate creates the role automatically. The role name in IAM is prefixed with `AmazonTranslateServiceRole-`.

**Note**
If you chose to encrypt your translation output with your own KMS key, then you cannot choose **Create an IAM role**. In this case, you must use a preexisting IAM role, and your KMS key must have a key policy that allows the role to use the key. For more information, see Prerequisite permissions to customize encryption (p. 32).

9. Choose **Create job**.

The console returns to the **Translation jobs** page, where the job creation status is shown in a banner at the top of the page. After a few minutes, your job is shown in the table.

10. Choose the job name in the **Name** column to open the job details page.

While your translation job runs, the **Status** field shows **In progress**.

11. When the status becomes **Completed**, go to your translation output by choosing the link under **Output file location**. The console goes to your output bucket in Amazon S3.

12. To download your output files, select the check box for each, and choose **Download**.

**AWS CLI**

To run a translation job by using the AWS CLI, use the `start-text-translation-job` command, and specify the name of your parallel data resource for the `parallel-data-names` parameter.

**Example Start-text-translation-job command**

The following example runs a translation job by submitting an Excel file that is stored in an input bucket in Amazon S3. This job is customized by the parallel data that is included in the request.

```bash
$ aws translate start-text-translation-job \
  --input-data-config ContentType=application/vnd.openxmlformats-officedocument.spreadsheetml.sheet,S3Uri=s3://my-s3-bucket/input/ \
  --output-data-config S3Uri=s3://my-s3-bucket/output/ \
  --data-access-role-arn arn:aws:iam::111122223333:role/my-iam-role \
  --source-language-code en \
  --target-language-codes es it \
  --job-name my-translation-job
```

If the command succeeds, Amazon Translate responds with the job ID and status:

```json
{
  "JobId": "4446f95f20c88a4b347449d3671fbe3d",
  "JobStatus": "SUBMITTED"
}
```

If you want to customize the output of your translation job, you can use the following parameters:

```
--settings
```

Settings to configure your translation output, including the following options:

Turn on brevity in the translation output. Amazon Translate doesn't support brevity for batch translation jobs. For more information, see Using brevity in Amazon Translate (p. 46).
Enable profanity to mask profane words and phrases. To enable, set the profanity parameter to Profanity=MASK. For more information, see Masking profane words and phrases in Amazon Translate (p. 47). If any of the target languages don't support profanity masking, the translation job won't mask profanity for any target language.

Set the level of formality in the translation output. Set the Formality parameter to FORMAL or INFORMAL. If you specify multiple target languages for the job, translate ignores the formality setting for any unsupported target language. For more information, see Setting formality in Amazon Translate (p. 48).

--terminology-names

The name of a custom terminology resource to add to the translation job. This resource lists example source terms and the desired translation for each term. If you specify multiple target languages for the job, translate uses the designated terminology for each requested target language that has an entry for the source term in the terminology file.

This parameter accepts only one custom terminology resource.

For a list of available custom terminology resources, use the list-terminologies command.

For more information, see Customizing your translations with custom terminology (p. 42).

--parallel-data-names

The name of a parallel data resource to add to the translation job. This resource consists of examples that show how you want segments of text to be translated. If you specify multiple target languages for the job, the parallel data file must include translations for all the target languages.

When you add parallel data to a translation job, you create an Active Custom Translation job.

This parameter accepts only one parallel data resource.

Note: Active Custom Translation jobs are priced at a higher rate than other jobs that don't use parallel data. For more information, see Amazon Translate pricing.

For a list of available parallel data resources, use the list-parallel-data command.

For more information, see Customizing your translations with parallel data (Active Custom Translation) (p. 50).

Example Describe-text-translation-job command

The following example checks the job status by providing the job ID. This ID was provided by Amazon Translate when the job was initiated by the start-text-translation-job command.

$ aws translate describe-text-translation-job \
> --job-id 4446f95f20c88a4b3474449d3671fbe3d

Amazon Translate responds with the job properties, which include its status:

```json
{
   "TextTranslationJobProperties": {
      "JobId": "4446f95f20c88a4b3474449d3671fbe3d",
      "JobName": "my-translation-job",
      "JobStatus": "COMPLETED",
      "JobDetails": {
         "TranslatedDocumentsCount": 0,
```
Monitoring and analyzing batch translation jobs

You can use a job’s ID to monitor its progress and get the Amazon S3 location of its output documents. To monitor a specific job, use the DescribeTextTranslationJob operation. You can also use the ListTextTranslationJobs operation to retrieve information on all of the translation jobs in your account. To restrict results to jobs that match a certain criteria, use the ListTextTranslationJobs operation’s filter parameter. You can filter results by job name, job status, or the date and time that the job was submitted.

Example describe-text-translation-job command

The following example checks a job’s status by using the AWS CLI to run the DescribeTextTranslationJob command:

```bash
$ aws translate describe-text-translation-job --job-id 1c1838f470806ab9c3e0057f14717bed
```

This command returns the following output:

```json
{
   "TextTranslationJobProperties": {
      "InputDataConfig": {
         "ContentType": "text/plain",
         "S3Uri": "s3://input-bucket-name/folder"
      },
      "EndTime": 1576551359.483,
      "SourceLanguageCode": "en",
      "DataAccessRoleArn": "arn:aws:iam::012345678901:role/service-role/AmazonTranslateInputOutputAccess",
      "JobId": "1c1838f470806ab9c3e0057f14717bed"
   }
}
```
You can stop a batch translation job while its status is IN_PROGRESS by using the StopTextTranslationJob operation.

**Example stop-text-translation-job command**

The following example stops a batch translation with by using the AWS CLI to run the StopTextTranslationJob command:

```bash
$ aws translate stop-text-translation-job --job-id 5236d36ce5192abdb3e2519f3ab8b065
```

This command returns the following output:

```json
{
  "TextTranslationJobProperties": {
    "InputDataConfig": {
      "ContentType": "text/plain",
      "S3Uri": "s3://input-bucket-name/folder"
    },
    "SourceLanguageCode": "en",
    "DataAccessRoleArn": "arn:aws:iam::012345678901:role/service-role/AmazonTranslateInputOutputAccess",
    "TargetLanguageCodes": [
      "fr"
    ],
    "JobName": "canceled-test",
    "SubmittedTime": 1576558958.167,
    "JobStatus": "STOP_REQUESTED",
    "JobId": "5236d36ce5192abdb3e2519f3ab8b065",
    "OutputDataConfig": {
      "S3Uri": "s3://output-bucket-name/012345678901-TranslateText-5236d36ce5192abdb3e2519f3ab8b065/"
    }
  }
}
```

**Getting batch translation results**

Once the job's status is COMPLETED or COMPLETED_WITH_ERROR, your output documents are available in the Amazon S3 folder you specified. The output document names match the input document names, with the addition of the target language code as a prefix. For instance, if you translated a document called mySourceText.txt into French, the output document will be called fr.mySourceText.txt.
If the status of a batch translation job is FAILED, the `DescribeTextTranslationJob` operation response includes a `Message` field that describes the reason why the job didn’t complete successfully.

Each batch translation job also generates an auxiliary file that contains information on the translations performed, such as the total number of characters translated and the number of errors encountered. This file, called `target-language-code`.auxiliary-translation-details.json, is generated in the details subfolder of your output folder.

The following is an example of a batch translation auxiliary file.

```json
{
  "sourceLanguageCode": "en",
  "targetLanguageCode": "fr",
  "charactersTranslated": "105",
  "documentCountWithCustomerError": "0",
  "documentCountWithServerError": "0",
  "inputDataPrefix": "s3://input-bucket-name/folder",
  "outputDataPrefix": "s3://output-bucket-name/012345678901-TranslateText-1c1838f470806ab9c3e0057f14717bed/",
  "details": [
    {
      "sourceFile": "mySourceText.txt",
      "targetFile": "fr.mySourceText.txt",
      "auxiliaryData": {
        "appliedTerminologies": [
          {
            "name": "TestTerminology",
            "terms": [
              {
                "sourceText": "Amazon",
                "targetText": "Amazon"
              }
            ]
          }
        ]
      }
    },
    {
      "sourceFile": "batchText.txt",
      "targetFile": "fr.batchText.txt",
      "auxiliaryData": {
        "appliedTerminologies": [
          {
            "name": "TestTerminology",
            "terms": [
              {
                "sourceText": "Amazon",
                "targetText": "Amazon"
              }
            ]
          }
        ]
      }
    }
  ]
}
```
Customizing your translations with Amazon Translate

You can use the following settings to customize the translations that you produce with Amazon Translate:

- **Do-not-translate tags** – Use start and end tags to specify content that you don’t want to translate (in HTML content).
- **Custom terminology** – Define how you want Amazon Translate to translate specific terms, such as brand names.
- **Brevity** – Reduces the length of the translation output for most translations (compared to the translation output without brevity). Use brevity with real-time text translations.
- **Profanity** – Mask profane words and phrases in your translation output.
- **Formality** – Set the level of language formality in your translation output.
- **Parallel data** – Adapt the translation output to reflect the style, tone, and word choices in the example translation samples that you provide.

**Topics**

- Using do-not-translate in Amazon Translate (p. 41)
- Customizing your translations with custom terminology (p. 42)
- Using brevity in Amazon Translate (p. 46)
- Masking profane words and phrases in Amazon Translate (p. 47)
- Setting formality in Amazon Translate (p. 48)
- Customizing your translations with parallel data (Active Custom Translation) (p. 50)

Using do-not-translate in Amazon Translate

For HTML content, you can add do-not-translate tags to text that you don’t want to translate. This feature is available for the console and API operations.

**Topics**

- Using do-not-translate with the console (p. 41)
- Using do-not-translate with the API (p. 42)

Using do-not-translate with the console

In the source HTML content, specify `translate="no"` in HTML tags that surround the content that you don’t want to translate. For example, to translate the following text from English to German:

In French, the Louvre Museum is Musée du Louvre.
The text "Musée du Louvre" needs to remain in French, so we use a span tag to skip translation for this content:

<p>In French, the Louvre Museum is <span translate="no">Musée du Louvre</span>.</p>

This sentence has the resulting translation into German:

<p>Auf Französisch ist <span translate="no">Musée du Louvre</span> das Louvre-Museum.</p>

### Using do-not-translate with the API

You can use do-not-translate with the real-time API operations (TranslateText and TranslateDocument) and the asynchronous StartTextTranslationJob API operation. In the source text that you provide for the API request, you can use any type of HTML element to specify content that needs to skip translation.

In the following example, we want to translate some text from English to Spanish, but keep some text in English:

```bash
aws translate translate-text \
  --source-language-code "en" \
  --target-language-code "es" \
  --region us-west-2 \
  --text "<p>You can translate this paragraph to any language.</p> <p translate=no>But do not translate this.</p>"
```

This API request returns the following Spanish translation:

```json
{
  "TranslatedText": "<p>Puede traducir este párrafo a cualquier idioma.</p> <p translate=no>But do not translate this.</p>",
  "SourceLanguageCode": "en",
  "TargetLanguageCode": "es"
}
```

### Customizing your translations with custom terminology

Use custom terminologies along with your translation requests to make sure that your brand names, character names, model names, and other unique content get translated to the desired result.

You can create terminology files and upload them to your Amazon Translate account. For information about file sizes and number of terminology files, see Service quotas (p. 118). When you translate text, you can optionally choose a custom terminology file to use. When Amazon Translate finds a match between source text and the terminology file, it uses the translation from the terminology file.

Consider the following example: Amazon Photos provides free photo and video storage to Amazon Prime members. In French, the name isn't translated: it remains as Amazon Photos.

When you use Amazon Translate to translate Amazon Photos into French without any additional context, the result is Photos d'Amazon, which isn't the desired translation.
If you add a custom terminology entry for the term *Amazon Photos*, specifying that the French translation is *Amazon Photos*, Amazon Translate uses the custom terminology to translate the phrase to the desired result.

Amazon Translate doesn't guarantee that it will use the target term for every translation. Custom terminology uses the meaning of the source and target term in the translation context to decide whether to use the target term. For more information, see [Best practices (p. 45)](#).

**Topics**
- Creating a custom terminology (p. 43)
- Using custom terminologies (p. 44)
- Encrypting your terminology (p. 45)
- Best practices (p. 45)

## Creating a custom terminology

You define custom terminology by creating a terminology file. Amazon Translate supports CSV, TSV, or TMX file formats. Each entry in the file contains the source term and the equivalent (translated) term for each target language.

After you create a terminology file, you upload the file to your Amazon Translate account.

**Important**

The source text in a custom terminology is **case-sensitive**. During translation, Amazon Translate uses the custom terminology when it finds an exact match in the input document.

### Terminology file formats

The following example shows a terminology file in CSV format.

**CSV (comma separated values)**

```
en,fr,es
Amazon Photos,Amazon Photos,Amazon Photos
```

The following example shows a terminology file in TMX format. A TMX file uses an XML format that translation software often uses.

**TMX (Translation Memory eXchange)**

```xml
<?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 xml:lang="en">
      <seg>Amazon Photos</seg>
    </tu>
    <tu xml:lang="fr">
      <seg>Amazon Photos</seg>
    </tu>
    <tu xml:lang="es">
      <seg>Amazon Photos</seg>
    </tu>
  </body>
</tmx>
```
Directionality

When you upload a custom terminology file, you set the *directionality* value for the custom terminology. Directionality indicates whether your terminology file specifies one source language or multiple source languages.

For directionality, set one of the following values:

**Uni-directional**

The terminology file contains one source language (the first language in the list). All other languages are target languages.

For example, in a CSV file, the first column contains text for the source language, and all other columns contain text for the target languages.

**Multi-directional**

Any language in the file can be a source language or a target language. For example, if your terminology file contains text in English, French, and Spanish, you can use the file for jobs that translate the following language pairs:

- English to French
- English to Spanish
- French to English
- French to Spanish
- Spanish to English
- Spanish to French

In contrast, you would need to create three uni-directional terminology files for these six translation jobs (one for each source language).

Using custom terminologies

To use a Custom Terminology when translating text with the `TranslateText` operation, include the optional `TerminologyNames` parameter.

For example, if you upload the following terminology file called `Amazon_Family.csv` to your account:

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

You can use the following CLI command to translate your text using Custom Terminology.

**Note**

This 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 (^).
Encrypting your terminology

Amazon Translate endeavors to protect all 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. AWS encryption is the default option to safeguard your information.
• Using an encryption key associated with your account. 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. The console displays an input field for you to enter the Amazon Resource Name (ARN) of the encryption key.

Best practices

Use following general best practices when using custom terminologies:

• Keep your custom terminology uncluttered. Only include terms for which you need to control the translated values.
• Custom terminologies are case-sensitive. If you want a target translation for the capitalized and non-capitalized versions of a word, include an entry for each version.
• Custom terminology isn't intended as a tool for controlling spacing, punctuation, or capitalization. For example, avoid the following types of entries:
  • Adding spaces – EN: USA FR: U S A
  • Adding punctuation – EN: USA FR: U.S.A
  • Changing the capitalization – EN: USA FR: Usa
• Don't include different translations for the same source phrase. For example:
  • Entry #1 – EN: Amazon FR: Amazon
• Entry #2 – EN: Amazon FR: Amazone
• You can create custom terminology files for any of the languages that Amazon Translate supports.

Amazon Translate doesn't guarantee that custom terminology will use the target term for every translation. To achieve high accuracy with custom terminology, follow these best practices when you create the content for the terminology file:

• Custom terminology works well with any words, including verbs and homographs. Proper names, such as brand names and product names, are ideal entries.
• Target terms should be fluent in the target language. Custom terminology isn't recommended for target terms that contain numerous special characters or formatting.
• You can include multi-word phrases or clauses in your terminology file. However, terms that contain multiple words are less likely to read fluently in the target languages.
• Custom terminology uses the meaning of the source and target term in the translation context to decide whether to use the target term. If a target term isn't a good fit in a given translation context, Amazon Translate may not use the target term.

For example, if your terminology file contains the following entry for English to French:

EN: order, FR: commande (based on the English "to order" translating into French "commander").

Amazon Translate doesn't use this entry when translating the following sentence, because the translation context doesn't match:

"In order for us to help you, please share your name."

Suggestions for avoiding this type of situation:
• Make sure that the target term for each language is semantically equivalent to the source term.
• Avoid source or target terms that have multiple meanings.

Using brevity in Amazon Translate

When translating between languages, there are times when the translation output is longer (in character count) than desired. Longer output can cause a problem in some scenarios (such as captions, subtitles, headlines, or form fields), if there is no space for extra characters.

You can turn on the brevity setting when you run real-time text translations with Amazon Translate. Brevity reduces the length of the translation output for most translations (compared to the translation output without brevity).

Topics
• Using the brevity setting (p. 46)
• Supported languages (p. 47)

Using the brevity setting

You can use the brevity setting with real-time text translation. Amazon Translate doesn't support brevity for real-time document translation or for asynchronous translation jobs.

To use brevity in a real-time text translation request, do one of the following:

• Console – In the Text tab of the Real-time translation page, under Additional settings, choose the Brevity setting.
• **AWS CLI** – In the `translate-text` command, set brevity in the `--settings` parameter to `Brevity=ON`. For more information, see `translate-text` in the *AWS CLI Command Reference*.

• **AWS API** – In the `TranslateText` API operation, configure brevity in the `Settings` parameter.

### Supported languages

Amazon Translate supports brevity for the following language pairs:

- From any [source language](p. 3) to one of the languages in the following table.
- From any of the languages in the following table to English.

<table>
<thead>
<tr>
<th>Language</th>
<th>Language code</th>
</tr>
</thead>
<tbody>
<tr>
<td>French</td>
<td>fr</td>
</tr>
<tr>
<td>German</td>
<td>de</td>
</tr>
<tr>
<td>Italian</td>
<td>it</td>
</tr>
<tr>
<td>Portuguese (Brazil)</td>
<td>pt</td>
</tr>
<tr>
<td>Spanish</td>
<td>es</td>
</tr>
</tbody>
</table>

If you request brevity for translation with an unsupported language pair, the translation proceeds with the brevity setting turned off.

### Masking profane words and phrases in Amazon Translate

When you run translations with Amazon Translate, you can enable the *profanity* setting to mask profane words and phrases in your translation output.

To mask profane words and phrases, Amazon Translate replaces them with the grawlix string “?$#@$”. This 5-character sequence is used for each profane word or phrase, regardless of the length or number of words.

Amazon Translate does not mask profanity in translation requests where the source language and target language are the same.

In some cases, a profane word in the source input might naturally become inoffensive in the translated output. In such cases, no masking is applied.

Amazon Translate detects each profane word or phrase literally, not contextually. This means that it might mask a profane word even if it’s inoffensive in context. For example, if Amazon Translate detected “jerk” as a profane word, then it would write the phrase “jerk chicken” as “?$#@$ chicken”, even though “jerk chicken” is inoffensive. (Here, “jerk” is used as an example only. Amazon Translate does not detect that word as profanity.)

### Topics

- **Using the profanity setting** (p. 48)
Using the profanity setting

You can use the profanity setting with both types of translation operations in Amazon Translate: real-time translation and asynchronous batch processing.

To mask profanity in a real-time translation request, do any of the following:

- On the Real-time translation page in the Amazon Translate console, under Additional settings, enable the Profanity setting.
- In the translate-text command in the AWS CLI, set the --settings parameter to Profanity=MASK. For more information, see translate-text in the AWS CLI Command Reference.
- In the TranslateText action in the Amazon Translate API, use the Settings parameter to set profanity masking. For more information, see TranslateText in the API Reference.

To mask profanity in an asynchronous batch operation, see Running a batch translation job (p. 34).

Unsupported languages

You can mask profanity when you translate to any of the target languages that Amazon Translate supports, with the following exceptions:

<table>
<thead>
<tr>
<th>Language</th>
<th>Language code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bengali</td>
<td>bn</td>
</tr>
<tr>
<td>Hindi</td>
<td>hi</td>
</tr>
<tr>
<td>Malayalam</td>
<td>ml</td>
</tr>
<tr>
<td>Punjabi</td>
<td>pa</td>
</tr>
<tr>
<td>Sinhala</td>
<td>si</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>vi</td>
</tr>
</tbody>
</table>

For all of the languages that Amazon Translate supports, see Supported languages and language codes (p. 3).

Setting formality in Amazon Translate

You can optionally specify the desired level of formality for translations to supported target languages. The formality setting controls the level of formal language usage (also known as honorifics or register) in the translation output. The formality setting is available for real-time translation and asynchronous batch processing.

Formality supports the following values:

- **Informal** – All sentences in the translated text use language constructs associated with informal communication. For example, translated text uses the familiar form of second person pronouns and their verb agreement (or Kudaketa form for Japanese).
• **Formal** – All sentences in the translated text use language constructs associated with formal, polite communication. For example, translated text uses the formal form of second person pronouns and their verb agreement (or Teineigo form for Japanese).

For example, the sentence ‘Are you sure?’ can have two correct translations in German: ‘Sind Sie sicher?’ for the formal register and ‘Bist du sicher?’ for the informal one.

If Amazon Translate doesn't support formality level for the target language, or you don't specify the formality parameter, the translation job ignores the formality setting.

**Topics**

- Using the formality setting (p. 49)
- Supported languages (p. 49)

**Using the formality setting**

To set formality in a real-time translation request, do one of the following:

- On the Real-time translation page in the Amazon Translate console, under Additional settings, enable the Formality setting and select one of the values.
- Use the Settings parameter in the TranslateText operation in the Amazon Translate API.
- For the translate-text command in the AWS CLI, set the --settings parameter to `Formality=FORMAL` or `Formality=INFORMAL`. For more information, see translate-text in the AWS CLI Command Reference.

To set formality in a batch translation request, set the Formality parameter when you start the translation job. For details and examples, see Running a batch translation job (p. 34).

For CLI or API requests, the AppliedSettings field in the response includes the formality setting (if any) from the request. If the target language doesn't support formality, the AppliedSettings value in the response is NULL.

**Supported languages**

Amazon Translate supports the formality setting for translation from any source language to the following target languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>Language code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dutch</td>
<td>nl</td>
</tr>
<tr>
<td>French</td>
<td>fr</td>
</tr>
<tr>
<td>French (Canada)</td>
<td>fr-CA</td>
</tr>
<tr>
<td>German</td>
<td>de</td>
</tr>
<tr>
<td>Hindi</td>
<td>hi</td>
</tr>
<tr>
<td>Italian</td>
<td>it</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja</td>
</tr>
<tr>
<td>Korean</td>
<td>ko</td>
</tr>
</tbody>
</table>
For all the languages that Amazon Translate supports, see Supported languages and language codes (p. 3).

Customizing your translations with parallel data (Active Custom Translation)

Add *parallel data* to Amazon Translate to customize the output of your batch translations jobs. Parallel data consists of examples that show how you want segments of text to be translated. It includes a collection of textual examples in a source language, and for each example, it contains the desired translation output in one or more target languages.

When you add parallel data to a batch translation job, you create an *Active Custom Translation* job. When you run these jobs, Amazon Translate uses your parallel data at runtime to produce customized machine translation output. It adapts the translation to reflect the style, tone, and word choices that it finds in your parallel data. With parallel data, you can tailor your translations for terms or phrases that are unique to a specific domain, such as life sciences, law or finance.

**Note**
Active Custom Translation jobs are priced at a higher rate than other jobs that don't use parallel data. For more information, see Amazon Translate pricing.

For example, the following parallel data is defined in a CSV file:

```
"en","fr"
"How are you?","Comment ça va ?"
```

In this example, English (en) is the source language, and French (fr) is the target language. The example shows how the source phrase "How are you?" should be translated into French. After this example input file is imported into Amazon Translate, it can be applied to translation jobs to influence their output. During such jobs, Amazon Translate translates "How are you?" into the informal "Comment ça va ?" as opposed to the formal "Comment allez-vous ?" For example, the job might receive the following source text:

Hello, how are you?
How are you?
Hi, how are you?
How are you doing?

From this text, the job produces the following translation:

```
Bonjour, comment ça va ?
Comment ça va ?
Salut, comment ça va ?
Comment ça va ?
```

In contrast, if the job runs without the parallel data, the output might include the more formal "comment allez-vous":

```
Comment allez-vous ?
Comment allez-vous ?
Comment allez-vous ?
Comment allez-vous ?
```

<table>
<thead>
<tr>
<th>Language</th>
<th>Language code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portuguese (Portugal)</td>
<td>pt-PT</td>
</tr>
<tr>
<td>Spanish</td>
<td>es</td>
</tr>
<tr>
<td>Spanish (Mexico)</td>
<td>es-MX</td>
</tr>
</tbody>
</table>
By customizing your batch translation jobs with parallel data, you influence the output in a way that's similar to using a custom translation model that you train with your translation examples. With Active Custom Translation, training a custom model is unnecessary, and you avoid the time and expense that such training requires. As your translation requirements change over time, you can refine your output by updating your parallel data, which is easier than retraining a custom model.

**Region availability**

Active Custom Translation is available in the following regions:

- US East (N. Virginia)
- US West (Oregon)
- Europe (Ireland)

**Topics**

- Parallel data input files for Amazon Translate (p. 51)
- Adding your parallel data to Amazon Translate (p. 55)
- Viewing and managing your parallel data in Amazon Translate (p. 57)

**Parallel data input files for Amazon Translate**

Before you can create a parallel data resource in Amazon Translate, you must create an input file that contains your translation examples. Your parallel data input file must use languages that Amazon Translate supports. For a list of these languages, see [Supported languages and language codes](#) (p. 3).

**Example parallel data**

The text in the following table provides examples of translation segments that can be formatted into a parallel data input file:

<table>
<thead>
<tr>
<th>en</th>
<th>es</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Translate is a neural machine translation</td>
<td>Amazon Translate es un servicio de traducción</td>
</tr>
<tr>
<td>service.</td>
<td>automática basado en redes neuronales.</td>
</tr>
<tr>
<td>Neural machine translation is a form of language</td>
<td>La traducción automática neuronal es una forma</td>
</tr>
<tr>
<td>translation automation that uses deep learning</td>
<td>de automatizar la traducción de lenguajes</td>
</tr>
<tr>
<td>models.</td>
<td>utilizando modelos de aprendizaje profundo.</td>
</tr>
<tr>
<td>Amazon Translate allows you to localize content</td>
<td>Amazon Translate le permite localizar contenido</td>
</tr>
<tr>
<td>for international users.</td>
<td>para usuarios internacionales.</td>
</tr>
</tbody>
</table>

The first row of the table provides the language codes. The first language, English (en), is the source language. Spanish (es) is the target language. The first column provides examples of source text. The other column contains examples of translations. When this parallel data customizes a batch job, Amazon Translate adapts the translation to reflect the examples.
Input file formats

Amazon Translate supports the following formats for parallel data input files:

- Translation Memory eXchange (TMX)
- Comma-separated values (CSV)
- Tab-separated values (TSV)

TMX

Example TMX input file

The following example TMX file defines parallel data in a format that Amazon Translate accepts. In this file, English (en) is the source language. Spanish (es) is the target language. As an input file for parallel data, it provides several examples that Amazon Translate can use to tailor the output of a batch job.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tmx version="1.4">
  <header srclang="en"/>
  <body>
    <tu>
      <tuv xml:lang="en">
        <seg>Amazon Translate is a neural machine translation service.</seg>
      </tuv>
      <tuv xml:lang="es">
        <seg>Amazon Translate es un servicio de traducción automática basado en redes neuronales.</seg>
      </tuv>
    </tu>
    <tu>
      <tuv xml:lang="en">
        <seg>Neural machine translation is a form of language translation automation that uses deep learning models.</seg>
      </tuv>
      <tuv xml:lang="es">
        <seg>La traducción automática neuronal es una forma de automatizar la traducción de lenguajes utilizando modelos de aprendizaje profundo.</seg>
      </tuv>
    </tu>
    <tu>
      <tuv xml:lang="en">
        <seg>Amazon Translate allows you to localize content for international users.</seg>
      </tuv>
      <tuv xml:lang="es">
        <seg>Amazon Translate le permite localizar contenido para usuarios internacionales.</seg>
      </tuv>
    </tu>
  </body>
</tmx>
```

TMX requirements

Remember the following requirements from Amazon Translate when you define your parallel data in a TMX file:

- Amazon Translate supports TMX 1.4b. For more information, see the TMX 1.4b specification on the Globalization and Localization Association website.
The header element must include the `srclang` attribute. The value of this attribute determines the source language of the parallel data.

The body element must contain at least one translation unit (tu) element.

Each tu element must contain at least two translation unit variant (tuv) elements. One of these `tuv` elements must have an `xml:lang` attribute that has the same value as the one assigned to the `srclang` attribute in the header element.

All `tuv` elements must have the `xml:lang` attribute.

All `tuv` elements must have a segment (seg) element.

While processing your input file, Amazon Translate skips certain `tu` or `tuv` elements if it encounters `seg` elements that are empty or contain only white space:

- If the `seg` element corresponds to the source language, Amazon Translate skips the `tu` element that the `seg` element occupies.
- If the `seg` element corresponds to a target language, Amazon Translate skips only the `tuv` element that the `seg` element occupies.

While processing your input file, Amazon Translate skips certain `tu` or `tuv` elements if it encounters `seg` elements that exceed 1000 bytes:

- If the `seg` element corresponds to the source language, Amazon Translate skips the `tu` element that the `seg` element occupies.
- If the `seg` element corresponds to a target language, Amazon Translate skips only the `tuv` element that the `seg` element occupies.

If the input file contains multiple `tu` elements with the same source text, Amazon Translate does one of the following:

- If the `tu` elements have the `changedate` attribute, it uses the element with the most recent date.
- Otherwise, it uses the element that occurs closest to the end of the file.

**CSV**

The following example CSV file defines parallel data in a format that Amazon Translate accepts. In this file, English (`en`) is the source language. Spanish (`es`) is the target language. As an input file for parallel data, it provides several examples that Amazon Translate can use to tailor the output of a batch job.

**Example CSV input file**

<table>
<thead>
<tr>
<th>en</th>
<th>es</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Translate is a neural machine translation service.</td>
<td>Amazon Translate es un servicio de traducción automática basado en redes neuronales.</td>
</tr>
<tr>
<td>Neural machine translation is a form of language translation automation that uses deep learning models.</td>
<td>La traducción automática neuronal es una forma de automatizar la traducción de lenguajes utilizando modelos de aprendizaje profundo.</td>
</tr>
<tr>
<td>Amazon Translate allows you to localize content for international users.</td>
<td>Amazon Translate le permite localizar contenido para usuarios internacionales.</td>
</tr>
</tbody>
</table>

**CSV requirements**

Remember the following requirements from Amazon Translate when you define your parallel data in a CSV file:

- The first row consists of the language codes. The first code is the source language, and each subsequent code is a target language.
- Each field in the first column contains source text. Each field in a subsequent column contains a target translation.
• If the text in any field contains a comma, the text must be enclosed in double quote (") characters.
• A text field cannot span multiple lines.
• Fields cannot start with the following characters: +, -, =, @. This requirement applies whether or not the field is enclosed in double quotes (").
• If the text in a field contains a double quote ("), it must be escaped with a double quote. For example, text such as:

```
34" monitor
```

Must be written as:

```
34"" monitor
```

• While processing your input file, Amazon Translate will skip certain lines or fields if it encounters fields that are empty or contain only white space:
  • If a source text field is empty, Amazon Translate skips the line that it occupies.
  • If a target translation field is empty, Amazon Translate skips only that field.
• While processing your input file, Amazon Translate skips certain lines or fields if it encounters fields that exceed 1000 bytes:
  • If a source text field exceeds the byte limit, Amazon Translate skips the line that it occupies.
  • If a target translation field exceeds the byte limit, Amazon Translate skips only that field.
• If the input file contains multiple records with the same source text, Amazon Translate uses the record that occurs closest to the end of the file.

TSV

The following example TSV file defines parallel data in a format that Amazon Translate accepts. In this file, English (en) is the source language. Spanish (es) is the target language. As an input file for parallel data, it provides several examples that Amazon Translate can use to tailor the output of a batch job.

**Example TSV input file**

```
en es
Amazon Translate is a neural machine translation service. Amazon Translate es un servicio de traducción automática basado en redes neuronales. Neural machine translation is a form of language translation automation that uses deep learning models. La traducción automática neuronal es una forma de automatizar la traducción de lenguajes utilizando modelos de aprendizaje profundo. Amazon Translate allows you to localize content for international users. Amazon Translate le permite localizar contenido para usuarios internacionales.
```

**TSV requirements**

Remember the following requirements from Amazon Translate when you define your parallel data in a TSV file:

• The first row consists of the language codes. The first code is the source language, and each subsequent code is a target language.
• Each field in the first column contains source text. Each field in a subsequent column contains a target translation.
• If the text in any field contains a tab character, the text must be enclosed in double quote (") characters.
A text field cannot span multiple lines.
Fields cannot start with the following characters: +, -, =, @. This requirement applies whether or not the field is enclosed in double quotes (").
If the text in a field contains a double quote ("), it must be escaped with a double quote. For example, text such as:

34" monitor

Must be written as:

34"" monitor

While processing your input file, Amazon Translate skips certain lines or fields if it encounters fields that are empty or contain only white space:
• If a source text field is empty, Amazon Translate skips the line that it occupies.
• If a target translation field is empty, Amazon Translate skips only that field.
While processing your input file, Amazon Translate skips certain lines or fields if it encounters fields that exceed 1000 bytes:
• If a source text field exceeds the byte limit, Amazon Translate skips the line that it occupies.
• If a target translation field exceeds the byte limit, Amazon Translate skips only that field.
• If the input file contains multiple records with the same source text, Amazon Translate uses the record that occurs closest to the end of the file.

Adding your parallel data to Amazon Translate

To add parallel data to Amazon Translate, you import a parallel data input file from Amazon S3. Afterwards, you can use the parallel data to customize the output produced by a batch translation job.

Prerequisites
Before you can add parallel data to Amazon Translate, you must:

• Have a parallel data input file. To create one, see Parallel data input files for Amazon Translate (p. 51).
• Have an Amazon S3 bucket in your AWS account. To create one, see How do I create an S3 Bucket? in the Amazon Simple Storage Service User Guide.
• Upload your input file to an Amazon S3 bucket. For more information, see How do I upload files and folders to an S3 bucket? in the Amazon Simple Storage Service User Guide.

Adding parallel data (Amazon Translate console)

To add parallel data by using the Amazon Translate console, use the Parallel data page:

1. Open the Amazon Translate console.
2. In the navigation menu on the left, choose Customization, and choose Parallel data.
3. On the Parallel data page, choose Create parallel data. The console shows the Create parallel data page.
4. Provide the following:

Name
A custom name for the parallel data resource. You must assign a name that is unique in the account and region.
Description - optional
A custom description.

Parallel data location on S3
The location of the parallel data input file in Amazon S3. To provide the location by navigating to the file in Amazon S3, choose Select file.

File format
The format of the parallel data input file. Supported formats are Translation Memory eXchange (TMX), comma-separated values (CSV), and tab-separated values (TSV).

5. Under Encryption key, choose an AWS KMS key to secure your parallel data. These KMS keys are managed by AWS Key Management Service (AWS KMS). For more information about AWS KMS, see the AWS Key Management Service Developer Guide.

Use AWS owned key
Use a KMS key that is owned and managed by Amazon Translate. This is the default option and is used to encrypt your information if you don’t choose another method. For more information, see AWS owned keys in the AWS Key Management Service Developer Guide.

Use key from current account
Use one of the KMS keys that you manage in AWS KMS in your AWS account. If you choose this option, a menu provides a list of your KMS keys to choose from. For more information, see Customer managed keys in the AWS Key Management Service Developer Guide.

Use key from different account
Use a KMS key that is managed in AWS KMS in a different AWS account. If you choose this option, the console provides a field for you to enter the Amazon Resource Name (ARN) of the KMS key.

For more information about encryption keys, see the AWS Key Management Service Developer Guide.

6. Choose Create parallel data.

The console returns to the Parallel data page, where the import status is shown in a banner at the top of the page. After a few minutes, your parallel data resource is shown in the table. When the value in the Status column is Active, the parallel data is ready for you to use in a batch translation job.

Error file for troubleshooting
If Amazon Translate generates any errors or warnings while processing your input file, the console provides an error file that you can download to review the error messages. The contents of this file resemble the following example:

```json
{
   "summary": {
      "record_error_count": 1,
      "record_skipped_count": 0
   },
   "messages": [
      {
         "content": "Number 1 TU element",
         "message": "Invalid TMX format. One tu element should contain exactly one tuv element with the source language code: en"
      }
   ]
}
```
Adding parallel data (AWS CLI)

To add parallel data by using the AWS CLI, use the `create-parallel-data` command.

**Example create-parallel-data command**

The following example creates a parallel data object by importing a TSV file from Amazon S3:

```
$ aws translate create-parallel-data \
  > --name my-parallel-data \
```

If the command succeeds, Amazon Translate responds with the status of the new parallel data object:

```
{
  "Name": "my-parallel-data",
  "Status": "CREATING"
}
```

You can monitor the ongoing status of the parallel data by using the `get-parallel-data` command. When the status is ACTIVE, the parallel data is ready for you to use in a batch translation job. For an example of the `get-parallel-data` command, see To view the details for a parallel data object (p. 59).

Using your parallel data

Now that you have created a parallel data resource, you can apply it to a batch translation job to customize the output. To run a batch job, see Running a batch translation job (p. 34).

Viewing and managing your parallel data in Amazon Translate

You can view all of the parallel data resources that you have added to Amazon Translate, and you can access detailed summaries for each one. As your translation requirements change, you can refine your translation output by updating your parallel data.

**Viewing and managing parallel data (Amazon Translate console)**

To view and manage your parallel data in the Amazon Translate console, use the Parallel data page:

**To view a list of your of parallel data resources**

1. Open the Amazon Translate console.
2. In the navigation menu on the left, choose Customization, and choose Parallel data. The table on this page lists the parallel data resources that you have added to Amazon Translate.

**To view the details for a parallel data resource**

- On the Parallel data page, choose the name of the parallel data resource in the Name column. The console opens the details page, which includes information such as the status, last updated date, source language, and target languages.
To update a parallel data resource

1. Upload the updated version of your parallel data as a new input file in an Amazon S3 bucket.
2. In the Amazon Translate console, go to the Parallel data page.
3. Select the parallel data that you want to update, and choose Update. The console shows the Update parallel data page.
4. Provide the following:
   - Description - optional
     An updated description.
   - Parallel data location on S3
     The location of the updated parallel data input file in Amazon S3. To provide the location by navigating to the file in Amazon S3, choose Select file.
   - Select parallel data file format
     The format of the parallel data input file. Supported formats are Translation Memory eXchange (TMX), comma-separated values (CSV), and tab-separated values (TSV).
5. Choose Save. Amazon Translate replaces the old parallel data with the new input file.

Viewing and managing parallel data (AWS CLI)

You can use the AWS CLI to view and update your parallel data resources.

To view a list of your parallel data resources

To view a list of the parallel data resources that you have added to Amazon Translate, use the list-parallel-data command.

Example list-parallel-data command

The following example returns a list of parallel data resources and their properties.

```
$ aws translate list-parallel-data
```

If the command succeeds, Amazon Translate returns an array like the following:

```
{
  "ParallelDataPropertiesList": [
    {
      "Name": "my-parallel-data",
      "Status": "ACTIVE",
      "SourceLanguageCode": "en",
      "TargetLanguageCodes": [
        "es",
        "ja",
        "zh"
      ],
      "ParallelDataConfig": {
        "S3Uri": "s3://input-bucket/parallel-data-file.tsv",
        "Format": "TSV"
      },
      "ImportedDataSize": 2283,
      "ImportedRecordCount": 3,
    }
  ]
}
```
To view the details for a parallel data object

To look up the details for a single parallel data resource, use the `get-parallel-data` command. This command returns the properties of the parallel data as well as a pre-signed S3 URL where you can download the input file that was used to create it.

Example get-parallel-data command

The following example gets the properties and download location for the `my-parallel-data` object:

```bash
$ aws translate get-parallel-data \
> --name my-parallel-data
```

If the command succeeds, Amazon Translate returns the properties and download location:

```
{
  "ParallelDataProperties": {
    "Name": "my-parallel-data",
    "Status": "ACTIVE",
    "SourceLanguageCode": "en",
    "TargetLanguageCodes": [
      "es",
      "ja",
      "zh"
    ],
    "ParallelDataConfig": {
      "S3Uri": "s3://input-bucket/parallel-data-file.tsv",
      "Format": "TSV"
    },
    "ImportedDataSize": 2283,
    "ImportedRecordCount": 3,
    "FailedRecordCount": 0,
    "CreatedAt": 1598597751.406,
    "LastUpdatedAt": 1598597911.675
  },
  "DataLocation": {
    "RepositoryType": "S3",
    "Location": "pre-signed S3 URL"
  }
}
```

To update a parallel data resource

To update a parallel data resource, first, upload a new input file to an Amazon S3 input bucket. Then, use the `update-parallel-data` command and specify the parallel data resource that you want to update. Amazon Translate replaces the old parallel data with the information that's in the new input file.

Example update-parallel-data command

The following command updates `my-parallel-data` with a new input file from Amazon S3:

```bash
$ aws translate update-parallel-data \
```
If the command succeeds, Amazon Translate provides a response like the following:

```
{
   "Name": "my-parallel-data",
   "Status": "ACTIVE",
   "LatestUpdateAttemptStatus": "UPDATING",
   "LatestUpdateAttemptAt": 1598601455.844
}
```

In this response, the `Status` field provides the status of the preexisting parallel data object, and the `LatestUpdateAttemptStatus` field provides the status of the current update attempt.
Examples

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

Topics
- Using Amazon Polly with Amazon Translate (p. 61)
- Using Amazon Translate to translate a chat channel (p. 65)
- Using Amazon Translate with Amazon DynamoDB (p. 73)
- Using Amazon Translate to translate a web page (p. 76)
- Using Amazon Translate to translate large documents (p. 79)
- Using Signature Version 4 with Amazon Translate (p. 81)

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 AWS Regions and Endpoints. For a list of supported regions for Amazon Polly, see AWS Regions and Endpoints in the AWS General Reference.
5. Make sure you have the minimum required permissions to run this example. For the required permissions policies, see Identity-based policies for Amazon Translate (p. 95) 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 user.

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.

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

[61]
<head>
<title>Amazon Translate Demo</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 align="left">
<th>Source Language Code:
<select id="sourceLanguageCodeDropdown">
<option value="en">en</option>
<option value="ar">ar</option>
<option value="cs">cs</option>
<option value="de">de</option>
<option value="es">es</option>
<option value="fr">fr</option>
<option value="it">it</option>
<option value="ja">ja</option>
<option value="pt">pt</option>
<option value="ru">ru</option>
<option value="tr">tr</option>
<option value="zh">zh</option>
<option value="zh-TW">zh-TW</option>
</select></th>
<th>Target Language Code:
<select id="targetLanguageCodeDropdown">
<option value="en">en</option>
<option value="ar">ar</option>
<option value="cs">cs</option>
<option value="de">de</option>
<option value="es">es</option>
<option value="fr">fr</option>
<option value="it">it</option>
<option value="ja">ja</option>
<option value="pt">pt</option>
<option value="ru">ru</option>
<option value="tr">tr</option>
<option value="zh">zh</option>
<option value="zh-TW">zh-TW</option>
</select></th>
</tr>
<tr><th><textarea id="inputText" name="inputText" rows="10" cols="50" placeholder="Text to translate..."></textarea></th><th><textarea id="outputText" name="outputText" rows="10" cols="50" placeholder="Translated text..."></textarea></th></tr>
<tr><th align="left">
<button type="button" name="translateButton" onclick="doTranslate()">Translate</button>
<button type="button" name="synthesizeButton" onclick="doSynthesizeInput()">Synthesize Input Speech</button>
<button type="button" name="clearButton" onclick="clearInputs()">Clear</button>
</th>
</tr>
</table>

</body>
```javascript
// set the focus to the input box
document.getElementById("inputText").focus();

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

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

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

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

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

    // the source language code and text to translate
    var sourceLanguage = "en";
    var sourceText = inputText;

    // translate the text
    translate.translateText({
        "SourceLanguageCode": sourceLanguage,
        "TargetLanguageCode": "es",
        "Text": sourceText
    }).promise().then(function(result) {
        var targetText = result.TranslatedText;

        // synthesize output
        polly.synthesizeSpeech({
            "LanguageCode": "es",
            "Text": targetText
        }).then(function(result) {
            var audioStream = result.AudioStream;
            var audioElement = document.getElementById("audioElement");
            audioElement.src = audioStream;
            audioElement.play();
        }).catch(function(err) {
            alert("Error synthesizing output speech: " + err);
        });
    }).catch(function(err) {
        alert("Error translating text: " + err);
    });
}
```
```javascript
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) {
        return;
    }
    var queryParams = {
        Text: text,
        VoiceId: voiceId
    }
    translate.synthesizeText(queryParams, function(err, data) {
        if (err) {
            console.log(err, err.stack);
            alert("Error calling Amazon Synthesize. " + err.message);
            return;
        }
        if (data) {
            outputTextArea.value = data.AudioString;
        }
    });
}
```
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 AWS Regions and Endpoints in the AWS General Reference.
5. Make sure you have the minimum required permissions to run this example. For the required permissions policies, see Identity-based policies for Amazon Translate (p. 95) 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 user.
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.

```html
<!doctype html>
<html lang="en">
<head>
<title>Amazon Translate</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<!-- Latest compiled and minified CSS for Bootstrap -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
<!-- Custom CSS -->
<style>
.topHeader
{
    background-color: #6441a4;
    padding: 10px;
    border-bottom: solid 1px #cacaca;
    color: white
}

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

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

body{
    margin-left: 0px;
    margin-right: 0px;
    height: 100%;
}
</style>
</head>
<body>
<div class="container-fluid">
```
Amazon Translate Developer Guide
Using Amazon Translate to translate a chat channel

<!--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>

/* Latest compiled and minified JavaScript */

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-QpG6A7M6bWtXVgR1s+GeLbbxJPSjAIrU70Vmuog8d格urh2+7rIYJ9c7vJ" crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>
<script src="aws-js-sdk/dist/aws-sdk-all.js"></script>
<script src="http://cdn.tmijs.org/js/1.2.1/tmi.min.js" integrity="sha384-eE0n7sm1W7D0U12XhS1qSpZTe6nupA00ovLfqEY0yVjtGRBmFsdmjbHyEYm6Bw" crossorigin="anonymous"></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
        };

        // In a real application, you need to include input validation and encoding of
        // the text message, to guard against
        // malicious attacks that inject javascript or other html/css tags in an attempt
        // to take over the browser.

        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******************************/
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.");
}

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);
            } else 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;
            }
        });
    }
}

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();
    });
}
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 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. 74), to a file named example.py. example.py is a Lambda function that calls the TranslateText 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. 75), 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.


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/.


13. On the function detail page, choose Test.

14. For Event name, type TestTranslate. For Configure test event, replace the JSON with the following:

```json
{
    "Speaker": "company"
}
```
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 results.

```
| "Hello 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 TranslateText 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 file called example.zip. Save the file in an S3 bucket in the same region that you are running the example.

```python
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']
```

---

74
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
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.

This example works on simple HTML pages without nested elements.

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. Make sure you have the minimum required permissions to run this example. For the required permissions policies, see Identity-based policies for Amazon Translate (p. 95).

```
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
```
4. Set up the credentials needed to run the sample. For instructions, see [Set up AWS Credentials and Region for Development](https://docs.aws.amazon.com/sdk-for-java/developerguide/creds.html) 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 [AWS Regions and Endpoints](https://aws.amazon.com/regions/) in the AWS General Reference.

```java
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());
        }
    }
}
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 quotas (p. 118). 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. Make sure you have the minimum required permissions to run this example. For the required permissions policies, see Identity-based policies for Amazon Translate (p. 95).
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 AWS Regions and Endpoints in the AWS General Reference.
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.

```java
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);
        }
    }
}
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 Signing AWS API requests in the AWS Identity and Access Management User Guide.

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 Install and configure the AWS Command Line Interface (AWS CLI) (p. 10).
2. Make sure that you have the minimum required permissions to run this example. For the required permissions policies, see Identity-based policies for Amazon Translate (p. 95).
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 AWS 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.

```python
# 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.

```python
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%d%H%M%S')
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 + '
' + canonical_uri + '
' + canonical_querystring + '
' + canonical_headers + '
' + signed_headers + '
' + 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 + '
' + amz_date + '
' + credential_scope + '
' + hashlib.sha256(canonical_request).hexdigest()

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

# Sign the string_to_sign using the signing_key.
signature = hmac.new(signed_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:
	' + request_parameters

response = requests.post(endpoint, data=request_parameters, headers=headers)
print 'Response:
	' + response.text
Tagging your resources

A tag is metadata that you can associate with an Amazon Translate resource. A tag consists of a key-value pair. You can add tags to **Parallel Data** and **Custom Terminology** resources.

Tags have two major functions: organizing your resources and providing tag-based access control. You can add tags to a resource and then create IAM policies to allow or restrict access to the resource based on its tags.

A policy can allow or disallow an operation based on the tags provided in your request (request-tags) or tags associated with the resource you're calling (resource-tags). For more information on using tags with IAM, see [Controlling access using tags](#) in the IAM User Guide.

Considerations for using tags with Amazon Translate:

- You can add up to 50 user tags per resource.
- You can add tags when you create the resource, or any time after you create it.
- A tag **key** is a required field but a tag **value** is optional.
- Tags don't have to be unique between resources, but the tags for a given resource must have unique keys.
- Tag keys and values are case sensitive.
- A tag key can have a maximum of 128 characters; a tag value can have a maximum of 256 characters.
- AWS system tags start with prefix `aws:` in the tag key or value. You can't add, edit, or delete tag names or values with this prefix. System tags are not included in your tags quota per resource.

**Note**

If you plan to use your tagging schema across multiple AWS services and resources, remember that other services may have different requirements for allowed characters.

**Topics**

- Tagging a new resource (p. 85)
- Viewing, updating, and deleting tags associated with a resource (p. 86)

Tagging a new resource

You can add tags to a **ParallelData** or **Custom Terminology** resource when you create it.

**To add tags to a new resource (console)**

1. Sign in to the Amazon Translate console.
2. From the left navigation pane, select the resource (**Parallel data** or **Custom terminology**) that you want to create.
3. Chose **Create parallel data** or **Create terminology**. The console displays the main 'create' page for your resource. At the end of this page, you see a 'Tags - optional' panel.
4. Choose **Add new tag** to add a tag for the resource. Enter a tag key and, optionally, a tag value.

5. Repeat step 4 until you have added all your tags. Each key must be unique for this resource.

6. Choose **Create parallel data** or **Create terminology** to create the resource.

You can also add tags using the Amazon Translate `CreateParallelData` API operation. The following example shows how to add tags with the create-parallel-data CLI command.

```
aws translate create-parallel-data \
  --name "myTest" \
  --parallel-data-config "{"format": "CSV", \
                  "S3Uri": "s3://test-input/TEST.csv"}" \
  --tags "{"Key": "color", "Value": "orange"}"
```

**Viewing, updating, and deleting tags associated with a resource**

You can view, update or delete the tags associated with a Parallel data or Custom terminology resource.
To update tags for an existing resource (console)

1. Sign in to the Amazon Translate console.
2. From the left navigation pane, select Parallel data or Custom terminology.
3. Select the resource that contains the tags you want to view, update, or delete. The console opens the details page for the resource.
4. Scroll down until you see the Tags panel. Here, you can see all the tags associated with your selected resource.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>orange</td>
</tr>
<tr>
<td>type</td>
<td>PDF</td>
</tr>
</tbody>
</table>

Select Manage tags to edit or remove tags from your resource.
5. Choose the text you want to modify, then edit your tag. You can also remove the tag by selecting Remove.
6. To add a new tag, select Add new tag, then enter the key and value in the blank fields.
7. When you're finished modifying your tags, select Save.
Security in Amazon Translate

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon Translate, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors, including the sensitivity of your data, your company’s requirements, and applicable laws and regulations.

This topic helps you understand how to apply the shared responsibility model when using AWS. The following topics show you how to configure AWS to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS resources.

**Topics**
- Data protection in Amazon Translate (p. 88)
- Identity and Access Management for Amazon Translate (p. 90)
- Monitoring Amazon Translate (p. 107)
- Compliance validation for Amazon Translate (p. 115)
- Resilience in Amazon Translate (p. 115)
- Infrastructure security in Amazon Translate (p. 115)
- Amazon Translate and interface VPC endpoints (AWS PrivateLink) (p. 116)

Data protection in Amazon Translate

Amazon Translate conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all of the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up roles with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
• Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon Simple Storage Service (Amazon S3).

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a Name field. This includes when you work with Amazon Translate or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Translate or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

Topics
• Encryption at rest (p. 89)
• Encryption in transit (p. 90)

Encryption at rest

For the batch translation jobs that you run with Amazon Translate, your translation input and output are both encrypted at rest. However, the encryption method is different for each.

Amazon Translate also uses an Amazon Elastic Block Store (Amazon EBS) volume encrypted with the default key.

Translation input

When you use Amazon Translate to translate documents in batch, you store a set of input documents in an Amazon S3 bucket. To encrypt these documents at rest, you can use the SSE-S3 server-side encryption option that is provided by Amazon S3. With this option, each object is encrypted with a unique key that is managed by Amazon S3.

For more information, see Protecting data using server-side encryption with Amazon S3-managed encryption keys (SSE-S3) in the Amazon Simple Storage Service User Guide.

Translation output

When Amazon Translate completes a batch translation job, it puts the output in an Amazon S3 bucket in your AWS account. To encrypt the output at rest, Amazon Translate uses the SSE-KMS encryption option that is provided by Amazon S3. With this option, your output is encrypted with a key that is stored in AWS Key Management Service (AWS KMS).

For more information about SSE-KMS, see Protecting Data using server-side encryption with AWS Key Management Service (SSE-KMS) in the Amazon Simple Storage Service User Guide.

For more information about KMS keys, see AWS KMS keys in the AWS Key Management Service Developer Guide.

For this encryption, Amazon Translate can use either of the following types of keys:

AWS managed key

By default, Amazon Translate uses an AWS managed key. This type of KMS key is created by AWS and stored in your account. However, you cannot manage this KMS key yourself. It is managed and used on your behalf only by AWS.
Customer managed key

Optionally, you can choose to encrypt your output with a customer managed key, which is a KMS key that you create, own, and manage in your AWS account.

Before you can use your own KMS key, you must add permissions to the IAM service role that Amazon Translate uses to access your output bucket in Amazon S3. If you want to use a KMS key that's in a different AWS account, you must also update the key policy in AWS KMS. For more information, see Prerequisite permissions to customize encryption (p. 32).

You can choose to use your customer managed key when you run a batch translation job. For more information, see Running a batch translation job (p. 34).

Encryption in transit

To encrypt data in transit, Amazon Translate uses TLS 1.2 with AWS certificates.

Identity and Access Management for Amazon Translate

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon Translate resources. IAM is an AWS service that you can use with no additional charge.

Topics
- Audience (p. 90)
- Authenticating with identities (p. 91)
- Managing access using policies (p. 93)
- How Amazon Translate works with IAM (p. 94)
- Identity-based policy examples for Amazon Translate (p. 99)
- AWS managed policies for Amazon Translate (p. 104)
- Troubleshooting Amazon Translate identity and access (p. 105)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon Translate.

Service user – If you use the Amazon Translate service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Translate features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Translate, see Troubleshooting Amazon Translate identity and access (p. 105).

Service administrator – If you're in charge of Amazon Translate resources at your company, you probably have full access to Amazon Translate. It's your job to determine which Amazon Translate features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the
basic concepts of IAM. To learn more about how your company can use IAM with Amazon Translate, see How Amazon Translate works with IAM (p. 94).

IAM administrator – If you’re an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Translate. To view example Amazon Translate identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon Translate (p. 99).

### Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company’s single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the AWS Sign-In User Guide.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don’t use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the IAM User Guide.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the AWS IAM Identity Center User Guide and Using multi-factor authentication (MFA) in AWS in the IAM User Guide.

### AWS account root user

When you create an AWS account, you begin with one 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 don’t use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the IAM User Guide.

### Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A federated identity is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the AWS IAM Identity Center User Guide.
IAM users and groups

An **IAM user** is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](https://docs.aws.amazon.com/IAM/latest/UserGuide/rotate_access_keys_for_long_term_credentials.html) in the IAM User Guide.

An **IAM group** is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named `IAMAdmins` and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user (instead of a role)](https://docs.aws.amazon.com/IAM/latest/UserGuide/when_to_use_iam.html) in the IAM User Guide.

IAM roles

An **IAM role** is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](https://docs.aws.amazon.com/IAM/latest/UserGuide/using-iam-roles.html) in the IAM User Guide.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Creating a role for a third-party Identity Provider](https://docs.aws.amazon.com/IAM/latest/UserGuide/creating-a-role-for-third-party-identity-provider.html) in the IAM User Guide. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets](https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-policy-language-permissions.html) in the AWS IAM Identity Center User Guide.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/using-iam-cross-account.html) in the IAM User Guide.

- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.

- **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, Resources, and Condition Keys for Amazon Translate](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference-iam-policy-language.html) in the Service Authorization Reference.

- **Service role** – A service role is an **IAM role** that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-service-role.html) in the IAM User Guide.
• **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

• **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 CLI or 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](https://docs.aws.amazon.com/IAM/latest/userguide/idp-use-permissions-saml.html) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role (instead of a user)](https://docs.aws.amazon.com/IAM/latest/userguide/idp-use-permissions-saml.html) in the *IAM User Guide*.

## Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](https://docs.aws.amazon.com/IAM/latest/userguide/overview-policies.html) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

### Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](https://docs.aws.amazon.com/IAM/latest/userguide/overview-policies.html) in the *IAM User Guide*.

Identity-based policies can be further categorized as **inline policies** or **managed policies**. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](https://docs.aws.amazon.com/IAM/latest/userguide/overview-policies.html) in the *IAM User Guide*.

### Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM **role trust policies** and Amazon S3 **bucket policies**. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.
Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

**Access control lists (ACLs)**

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list (ACL) overview](https://docs.aws.amazon.com/AmazonS3/latest/dev/access-control-lists.html) in the *Amazon Simple Storage Service Developer Guide*.

**Other policy types**

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the **Principal** field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see *Permissions boundaries for IAM entities* in the *IAM User Guide*.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see *How SCPs work* in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see *Session policies* in the *IAM User Guide*.

**Multiple policy types**

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see *Policy evaluation logic* in the *IAM User Guide*.

**How Amazon Translate works with IAM**

Before you use IAM to manage access to Amazon Translate, learn what IAM features are available to use with Amazon Translate.

**IAM features you can use with Amazon Translate**

<table>
<thead>
<tr>
<th>IAM feature</th>
<th>Amazon Translate support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity-based policies (p. 95)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Identity-based policies for Amazon Translate

Supports identity-based policies  Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the IAM User Guide.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the IAM User Guide.

Identity-based policy examples for Amazon Translate

To view examples of Amazon Translate identity-based policies, see Identity-based policy examples for Amazon Translate (p. 99).

Resource-based policies within Amazon Translate

Supports resource-based policies  No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.
To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the IAM User Guide.

Policy actions for Amazon Translate

<table>
<thead>
<tr>
<th>Supports policy actions</th>
<th>Yes</th>
</tr>
</thead>
</table>

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as permission-only actions that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called dependent actions.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Translate actions, see Actions Defined by Amazon Translate in the Service Authorization Reference.

Policy actions in Amazon Translate use the following prefix before the action:

```
translate
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
    "translate:ListLanguages",
    "translate:TranslateText"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word List, include the following action:

```
"Action": "translate:List*"
```

Don't use wildcards to specify all of the actions for a service. Use the best practice of granting least privilege when you specify the permissions in a policy.

To view examples of Amazon Translate identity-based policies, see Identity-based policy examples for Amazon Translate (p. 99).

Policy resources for Amazon Translate

<table>
<thead>
<tr>
<th>Supports policy resources</th>
<th>Yes</th>
</tr>
</thead>
</table>
Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as resource-level permissions.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

"Resource": "*"

To see a list of Amazon Translate resource types and their ARNs, see Resources Defined by Amazon Translate in the Service Authorization Reference. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon Translate.

For examples of how to use resources in Amazon Translate policies, see Specify resources in a policy (p. 101).

Policy condition keys for Amazon Translate

| Supports service-specific policy condition keys | Yes |

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Condition element (or Condition block) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the IAM User Guide.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the IAM User Guide.

To see a list of Amazon Translate condition keys, see Condition Keys for Amazon Translate in the Service Authorization Reference. To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon Translate.

To view examples of Amazon Translate identity-based policies, see Identity-based policy examples for Amazon Translate (p. 99).

ACLs in Amazon Translate

| Supports ACLs | No |
Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

### ABAC with Amazon Translate

#### Supports ABAC (tags in policies)

Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the *condition element* of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is *Yes* for the service. If a service supports all three condition keys for only some resource types, then the value is *Partial*.

For more information about ABAC, see *What is ABAC?* in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see *Use attribute-based access control (ABAC)* in the *IAM User Guide*.

For more information about tagging Amazon Translate resources, see *Tagging your resources* (p. 85).

### Using temporary credentials with Amazon Translate

#### Supports temporary credentials

Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see *AWS services that work with IAM* in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company’s single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see *Switching to a role (console)* in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see *Temporary security credentials in IAM*.

### Cross-service principal permissions for Amazon Translate

#### Supports principal permissions

Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that
then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for Amazon Translate in the Service Authorization Reference.

### Service roles for Amazon Translate

<table>
<thead>
<tr>
<th>Supports service roles</th>
<th>Yes</th>
</tr>
</thead>
</table>

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the IAM User Guide.

**Warning**

Changing the permissions for a service role might break Amazon Translate functionality. Edit service roles only when Amazon Translate provides guidance to do so.

To use the Amazon Translate asynchronous operations, you must grant Amazon Translate access to the Amazon S3 bucket that contains your input documents. You do this by creating a service role in your account with a trust policy to trust the Amazon Translate service principal.

For a policy example, see Prerequisites for batch translation jobs (p. 30).

### Service-linked roles for Amazon Translate

<table>
<thead>
<tr>
<th>Supports service-linked roles</th>
<th>No</th>
</tr>
</thead>
</table>

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the Service-linked role column. Choose the Yes link to view the service-linked role documentation for that service.

### Identity-based policy examples for Amazon Translate

By default, users and roles don’t have permission to create or modify Amazon Translate resources. They also can’t perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant permission to perform specific API operations on the specific resources that they need. The administrator must then attach those policies to the users or roles that require those permissions.

To learn how to create an IAM identity-based policy using the following example JSON policy documents, see Creating Policies on the JSON Tab in the IAM User Guide.

**Topics**

- Identity-based policy best practices (p. 100)
- Allow access to the Amazon Translate console (p. 100)
- Allow users to view their own permissions (p. 101)
- Specify resources in a policy (p. 101)
- Permissions for using customer managed keys with custom terminologies (p. 102)
Identity-based policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon Translate resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the AWS managed policies that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](https://docs.aws.amazon.com/IAM/latest/userguide/policies-managed.html) or [AWS managed policies for job functions](https://docs.aws.amazon.com/IAM/latest/userguide/policies-managed-job-functions.html) in the IAM User Guide.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as least-privilege permissions. For more information about using IAM to apply permissions, see [Policies and permissions in IAM](https://docs.aws.amazon.com/IAM/latest/userguide/policies-basics.html) in the IAM User Guide.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see [IAM JSON policy elements: Condition](https://docs.aws.amazon.com/IAM/latest/userguide/iam-policies-conditions.html) in the IAM User Guide.

- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [IAM Access Analyzer policy validation](https://docs.aws.amazon.com/IAM/latest/userguide/iam-access-analyzer.html) in the IAM User Guide.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Configuring MFA-protected API access](https://docs.aws.amazon.com/IAM/latest/userguide/using-mfa.html) in the IAM User Guide.

For more information about best practices in IAM, see [Security best practices in IAM](https://docs.aws.amazon.com/IAM/latest/userguide/s保管最佳实践.html) in the IAM User Guide.

Allow access to the Amazon Translate console

To access the Amazon Translate console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Translate resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users, groups or roles) with that policy.

For Amazon Translate console permissions, you can attach the `TranslateFullAccess` AWS managed policy to the entities. For more information, see [AWS managed policies for Amazon Translate](https://docs.aws.amazon.com/IAM/latest/userguide/policies-managed.html#translatefullaccess) (p. 104).

You also need permissions for the actions shown in the following policy. These permissions are included in the `TranslateFullAccess` policy.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "iam:ListRoles",
```
Identity-based policy examples

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform. For more information, see Adding Permissions to a User in the IAM User Guide.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Specify resources in a policy

For many Amazon Translate API actions, you can restrict the scope of a policy by specifying resources that are allowed (or not allowed) for the action. For a list of the actions that can specify resources, see Actions Defined by Amazon Translate. You can specify the following resources in a policy:

- **Custom terminology** – Use the following ARN format:
Identity-based policy examples

- **Parallel data** – Use the following ARN format:


You can use the wildcard character to specify multiple resources in the policy. The following example policy allows all custom terminology resources for all Amazon Translate actions.

**Example**

```json
{
  "Sid": "Example1",
  "Effect": "Allow",
  "Action": "translate:*",
  "Resource": [
  ]
}
```

The following example policy denies access to a specific parallel data resource for the `GetParallelData` action.

**Example**

```json
{
  "Sid": "Example2",
  "Effect": "Deny",
  "Action": "translate:GetParallelData",
  "Resource": [
  ]
}
```

**Permissions for using customer managed keys with custom terminologies**

If you use AWS Key Management Service (AWS KMS) customer managed keys with Amazon Translate custom terminologies, you might need additional permissions in your KMS key policy.

To call the `ImportTerminology` operation with a customer managed key, add the following permissions to your existing KMS key policy.

```json
{
  "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:CreateAlias",
        "kms:CreateGrant",
        "kms:DescribeKey",
      ]
    }
  ]
}
```
To call the `GetTerminology` operation for a custom terminology that was imported with a KMS customer managed key, add the following permissions in the KMS key 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",
        "kms:GetKeyPolicy",
        "kms:PutKeyPolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

To call the `ListTerminologies` or `DeleteTerminology` operations for a custom terminology that was imported with a customer managed key, you don't need to have any special AWS KMS permissions.

To use customer managed keys with all custom terminologies operations, add the following permissions in the KMS key 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:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:GetKeyPolicy",
        "kms:PutKeyPolicy",
        "kms:RetireGrant"
      ],
      "Resource": "*"
    }
  ]
}
```
For details about the Amazon Translate operations and resources, see Actions, resources, and condition keys for Amazon Translate in the Service Authorization Reference.

AWS managed policies for Amazon Translate

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started, you can use AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the IAM User Guide.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the ReadOnlyAccess AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the IAM User Guide.

Topics

- AWS managed policy: TranslateFullAccess (p. 104)
- AWS managed policy: TranslateReadOnly (p. 105)
- Amazon Translate updates to AWS managed policies (p. 105)

AWS managed policy: TranslateFullAccess

This policy grants full access to Amazon Translate resources, the Amazon Comprehend DetectDominantLanguage API operation, and required CloudWatch API operations. The policy also grants list and get permissions for Amazon S3 buckets and IAM roles.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "translate:*",
                "comprehend:DetectDominantLanguage",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "iam:ListRoles",
                "iam:GetRole"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```
AWS managed policy: TranslateReadOnly

This policy grants permission to access the Amazon Translate API operations that do not modify resources associated with your account. The policy also grants permission to access the Amazon Comprehend DetectDominantLanguage API operation and required CloudWatch API operations.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "translate:TranslateText",
            "translate:TranslateDocument",
            "translate:GetTerminology",
            "translate:ListTerminologies",
            "translate:ListTextTranslationJobs",
            "translate:DescribeTextTranslationJob",
            "translate:GetParallelData",
            "translate:ListParallelData",
            "comprehend:DetectDominantLanguage",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:ListMetrics"
         ],
         "Effect": "Allow",
         "Resource": "*"
      }
   ]
}
```

Amazon Translate updates to AWS managed policies

View details about updates to AWS managed policies for Amazon Translate since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon Translate Document history page.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>TranslateReadOnly (p. 105) – Update to an existing policy</td>
<td>Amazon Translate now allows the TranslateDocument action in the TranslateReadOnly policy</td>
<td>May 23, 2023</td>
</tr>
<tr>
<td>Amazon Translate started tracking changes</td>
<td>Amazon Translate started tracking changes for its AWS managed policies.</td>
<td>May 23, 2023</td>
</tr>
</tbody>
</table>

Troubleshooting Amazon Translate identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Translate and IAM.

Topics
• I am not authorized to perform an action in Amazon Translate (p. 106)
• I am not authorized to perform iam:PassRole (p. 106)
• I want to allow people outside of my AWS account to access my Amazon Translate resources (p. 106)

I am not authorized to perform an action in Amazon Translate

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a fictional *my-example-widget* resource but does not have the fictional translate:*GetWidget* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: translate:GetWidget on resource: my-example-widget
```

In this case, Mateo's policy must be updated to allow him to access the *my-example-widget* resource using the translate:*GetWidget* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the *iam:PassRole* action, your policies must be updated to allow you to pass a role to Amazon Translate.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named marymajor tries to use the console to perform an action in Amazon Translate. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the *iam:PassRole* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon Translate resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

• To learn whether Amazon Translate supports these features, see How Amazon Translate works with IAM (p. 94).
• To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the IAM User Guide.
• To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the IAM User Guide.

• To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the IAM User Guide.

• To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.

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. To get different views of how your solution is performing over time, you can change the time range that the graphs show.

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.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful request count</td>
<td>The number of successful requests made to Amazon Translate during the specified time period.</td>
</tr>
<tr>
<td>Throttled request count</td>
<td>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 requests to Amazon Translate too quickly.</td>
</tr>
<tr>
<td>Graph</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Average response time (milliseconds)</strong>&lt;br&gt;<img src="image1.png" alt="Graph" /></td>
<td><strong>Average response time</strong>&lt;br&gt;The average length of time that it took Amazon Translate to process your request during the specified time period.</td>
</tr>
<tr>
<td><strong>Character count</strong>&lt;br&gt;<img src="image2.png" alt="Graph" /></td>
<td><strong>Character count</strong>&lt;br&gt;The total number of characters that you sent to Amazon Translate during the specified time period. This is the number of characters that you will be billed for.</td>
</tr>
<tr>
<td><strong>User error count</strong>&lt;br&gt;<img src="image3.png" alt="Graph" /></td>
<td><strong>User error count</strong>&lt;br&gt;The number of user errors that occurred during the specified time period. User errors are in the HTTP error code range 400-499.</td>
</tr>
<tr>
<td><strong>System error count</strong>&lt;br&gt;<img src="image4.png" alt="Graph" /></td>
<td><strong>System error count</strong>&lt;br&gt;The number of system errors that occurred during the specified time period. System errors are in the HTTP error code range 500-599.</td>
</tr>
</tbody>
</table>
Monitoring Amazon Translate

With Amazon 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 solutions 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.

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. 111).

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.

<table>
<thead>
<tr>
<th>How do I?</th>
<th>Monitor this metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track the number of successful requests</td>
<td>The sum statistic of the SuccessfulRequestCount metric</td>
</tr>
<tr>
<td>Know if my application has reached its maximum throughput</td>
<td>The sum statistic of the ThrottledCount metric</td>
</tr>
<tr>
<td>Find the response time for my application</td>
<td>The average statistic of the ResponseTime metric</td>
</tr>
<tr>
<td>Find the number of errors for my application</td>
<td>The sum statistic of the ServerErrorCount and UserErrorCount metrics</td>
</tr>
<tr>
<td>Find the number of billable characters</td>
<td>The sum statistic of the CharacterCount metric</td>
</tr>
</tbody>
</table>

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.

Logging Amazon Translate API calls with AWS CloudTrail

Amazon Translate is integrated with AWS CloudTrail, a service that provides a record of actions taken by an IAM entity or AWS service in Amazon Translate. CloudTrail captures all API calls for Amazon Translate as events. This includes calls from the Amazon Translate console and code calls to the Amazon Translate API operations. If you create a CloudTrail trail, you can enable continuous delivery of CloudTrail events, including events for Amazon Translate, to an Amazon Simple Storage Service (Amazon S3) bucket. If you don’t configure a trail, you can still view the most recent events in the CloudTrail console in Event history. You can use the information collected by CloudTrail to determine the request that was made to Amazon Translate, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Topics
- Amazon Translate information in CloudTrail (p. 110)
- Understanding Amazon Translate log file entries (p. 111)

Amazon Translate information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon Translate, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon Translate, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail with the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the S3 bucket that you specify. You can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail Supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All Amazon Translate actions are logged by CloudTrail and are documented in the API reference section. For example, calls to the DeleteTerminology, ImportTerminology and TranslateText actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. This information helps you determine the following:

- Whether the request was made with the root user credentials
- Whether the request was made with temporary security credentials for a role or federated user
• Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity element.

Understanding Amazon Translate log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren’t an ordered stack trace of the public API calls, so they don’t appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the TranslateText action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/Administrator",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "Administrator"
  },
  "eventTime": "2019-09-03T20:32:50Z",
  "eventSource": "translate.amazonaws.com",
  "eventName": "TranslateText",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/1.16.207 Python/3.4.7 Linux/4.9.184-0.1.ac.235.83.329.metall.x86_64 botocore/1.12.197",
  "requestParameters": {
    "text": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "sourceLanguageCode": "en",
    "targetLanguageCode": "fr"
  },
  "responseElements": {
    "translatedText": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "sourceLanguageCode": "en",
    "targetLanguageCode": "fr"
  },
  "requestID": "f56da956-28ae-4938-b6fc-59befa20e2bf",
  "eventID": "1dc75278-84d7-4bb2-861a-493d08d67391",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

CloudWatch metrics and dimensions for Amazon Translate

To monitor your solution’s performance, use the Amazon CloudWatch metrics and dimensions for Amazon Translate.

CloudWatch Metrics for Amazon Translate

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharacterCount</td>
<td>The number of billable characters in requests.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ResponseTime</strong></td>
<td>The time that it took to respond to a request.</td>
</tr>
<tr>
<td></td>
<td>Valid dimensions: Language pair, Operation</td>
</tr>
<tr>
<td></td>
<td>Valid statistics: Data samples, Average</td>
</tr>
<tr>
<td></td>
<td>Unit: For Data samples, count. For Average statistics, milliseconds.</td>
</tr>
<tr>
<td><strong>ServerErrorCount</strong></td>
<td>The number of server errors. The HTTP response code range for a server error is 500 to 599.</td>
</tr>
<tr>
<td></td>
<td>Valid dimension: Operation</td>
</tr>
<tr>
<td></td>
<td>Valid statistics: Average, Sum</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td><strong>SuccessfulRequestCount</strong></td>
<td>The number of successful translation requests. The response code for a successful request is 200 to 299.</td>
</tr>
<tr>
<td></td>
<td>Valid dimension: Operation</td>
</tr>
<tr>
<td></td>
<td>Valid statistics: Average, Sum</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td><strong>ThrottledCount</strong></td>
<td>The number of requests subject to throttling. Use ThrottledCount 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 Amazon Web Services General Reference.</td>
</tr>
<tr>
<td></td>
<td>Valid dimension: Operation</td>
</tr>
<tr>
<td></td>
<td>Valid statistics: Average, Sum</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td><strong>UserErrorCount</strong></td>
<td>The number of user errors that occurred. The HTTP response code range for a user error is 400 to 499.</td>
</tr>
<tr>
<td></td>
<td>Valid dimension: Operation</td>
</tr>
<tr>
<td></td>
<td>Valid statistics: Average, Sum</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
</tbody>
</table>

**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.
Monitoring Amazon Translate events with Amazon EventBridge

Amazon Translate integrates with Amazon EventBridge to notify you about changes that affect your translation jobs and parallel data resources. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you, and what automated actions to take when an event matches a rule. For example, actions that can be automatically started include:

- Invoking an AWS Lambda function
- Invoking AWS Systems Manager Run Command
- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine
- Notifying an Amazon SNS topic or an Amazon SQS queue

For more information, see Creating Amazon EventBridge rules that react to events in the Amazon EventBridge User Guide.

Amazon Translate events

The following are example events from Amazon Translate.

Events for batch translation jobs

You run batch translation jobs by using the Amazon Translate console or the StartTextTranslationJob operation. Amazon Translate sends events when these jobs are complete, either successfully or unsuccessfully. These events resemble the following example.

```json
{
    "version": "0",
    "id": "CWE-event-id",
    "detail-type": "Translate TextTranslationJob State Change",
    "source": "aws.translate",
    "account": "111122223333",
    "time": "2017-04-22T03:31:47Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobId": "01234567-0123-0123-0123-012345678901",
        "jobStatus": "STATUS"
    }
}
```

The value for the jobStatus attribute depends on the job state that Amazon Translate sent the event for. The jobStatus values are:

- COMPLETED – The job has successfully completed and the output is available.
• COMPLETED_WITH_ERROR – The job has completed with errors. The errors can be analyzed in the job’s output.
• STOPPED – The job has been stopped.
• FAILED – The job did not complete. To get details, use the DescribeTextTranslationJob operation.

Events for parallel data resources

When you use Amazon Translate to create or update a parallel data resource, it sends an event to indicate whether the operation succeeded or failed.

You create parallel data resources by using the Amazon Translate console or the CreateParallelData operation. When you do this, Amazon Translate sends an event like the following.

```json
{
    "version": "0",
    "id": "CWE-event-id",
    "detail-type": "Translate Parallel Data State Change",
    "source": "aws.translate",
    "account": "111122223333",
    "time": "2017-04-22T03:31:47Z",
    "region": "us-east-1",
    "resources": [arn:aws:translate:us-east-1:111122223333:parallel-data/ExampleParallelData],
    "detail": {
        "operation": "CreateParallelData",
        "name": "ExampleParallelData",
        "status": "ACTIVE"
    }
}
```

Values for the status attribute are:

• ACTIVE – The CreateParallelData operation succeeded, and the resource is ready for you to use.
• FAILED – The CreateParallelData operation failed.

You update parallel data resources by using the Amazon Translate console or the UpdateParallelData operation. When you do this, Amazon Translate sends an event like the following.

```json
{
    "version": "0",
    "id": "CWE-event-id",
    "detail-type": "Translate Parallel Data State Change",
    "source": "aws.translate",
    "account": "111122223333",
    "time": "2017-04-22T03:31:47Z",
    "region": "us-east-1",
    "resources": [arn:aws:translate:us-east-1:111122223333:parallel-data/ExampleParallelData],
    "detail": {
        "operation": "UpdateParallelData",
        "name": "ExampleParallelData",
        "status": "ACTIVE",
        "latestUpdateAttemptStatus": "ACTIVE",
        "latestUpdateAttemptAt": "2017-04-22T03:31:47Z"
    }
}
```

The status attribute provides the status of the prior version of the parallel data resource, which is being replaced by the update. Values are:
• ACTIVE – The prior version was created or updated successfully.
• FAILED – The prior version failed to be created or updated.

The latestUpdateAttemptStatus attribute provides the status of the new version of the parallel data resource, which is being created by the update. Values are:
• ACTIVE – The UpdateParallelData operation succeeded, and the updated resource is ready for you to use.
• FAILED – The UpdateParallelData operation failed.

Compliance validation for Amazon Translate

Third-party auditors assess the security and compliance of Amazon Translate as part of multiple AWS compliance programs. These include PCI, FedRAMP, HIPAA, and others. You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using Amazon Translate is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

• Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
• Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
• AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
• AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
• AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

Resilience in Amazon Translate

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in Amazon Translate

As a managed service, Amazon Translate is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.
To access Amazon Translate through the network, you use AWS published API calls. Clients must support TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems, such as Java 7 and later, support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an AWS Identity and Access Management (IAM) principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Amazon Translate and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon Translate by creating an interface VPC endpoint. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access Amazon Translate APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with Amazon Translate APIs. Traffic between your VPC and Amazon Translate does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

Considerations for Amazon Translate VPC endpoints

Before you set up an interface VPC endpoint for Amazon Translate, ensure that you review Interface endpoint properties and limitations in the Amazon VPC User Guide.

Amazon Translate supports making calls to all of its API actions from your VPC.

Creating an interface VPC endpoint for Amazon Translate

You can create a VPC endpoint for the Amazon Translate service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the Amazon VPC User Guide.

Create a VPC endpoint for Amazon Translate using the following service name:

• com.amazonaws.region.translate

If you enable private DNS for the endpoint, you can make API requests to Amazon Translate using its default DNS name for the Region, for example, translate.us-east-1.amazonaws.com.

For more information, see Accessing a service through an interface endpoint in the Amazon VPC User Guide.

Creating a VPC endpoint policy for Amazon Translate

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon Translate. The policy specifies the following information:
The principal that can perform actions.
• The actions that can be performed.
• The resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the Amazon VPC User Guide.

Example: VPC endpoint policy for Amazon Translate real-time translation actions

The following is an example of an endpoint policy for real-time translation in Amazon Translate. When attached to an endpoint, this policy grants access to the listed Amazon Translate actions for all principals on all resources.

```json
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": ["translate:TranslateText"],
            "Resource": "*"
        }
    ]
}
```

Example: VPC endpoint policy for Amazon Translate batch translation actions

The following is an example of an endpoint policy for batch translation in Amazon Translate. When attached to an endpoint, this policy grants access to the listed Amazon Translate actions for all principals on all resources.

```json
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": ["translate:StartTextTranslationJob", "iam:PassRole"],
            "Resource": "*"
        }
    ]
}
```
Guidelines and quotas

The following sections contain information about Amazon Translate guidelines and quotas.

**Topics**
- Supported AWS Regions (p. 118)
- Compliance (p. 118)
- Throttling (p. 118)
- Guidelines (p. 118)
- Service quotas (p. 118)

**Supported AWS Regions**

For a list of AWS Regions that support Amazon Translate, see [Amazon Translate endpoints and quotas](#) in the *AWS General Reference*.

**Compliance**

For more information about Amazon Translate compliance programs, see [AWS Compliance](#), [AWS Compliance Programs](#), and [AWS Services in Scope by Compliance Program](#).

**Throttling**

Amazon Translate scales to serve customer operational traffic. If you encounter sustained throttling, contact [AWS Support](#).

**Guidelines**

To continuously improve the quality of its analysis models, Amazon Translate might store your data. To learn more, see the [Amazon Translate FAQ](#).

You can request that we delete your data and that future data associated with your account isn't stored by contacting [AWS Support](#). However, because deleting your data can also delete unique training data that is helpful in improving translation, doing so might reduce the quality of your translations.

**Service quotas**

Amazon Translate has the following service guidelines and quotas.
### Synchronous real-time translation quotas

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Maximum input text</td>
<td>10,000 bytes</td>
</tr>
<tr>
<td>Maximum number of characters per document</td>
<td>100,000</td>
</tr>
<tr>
<td>Maximum document size</td>
<td>100,000 bytes</td>
</tr>
</tbody>
</table>

### Asynchronous batch translation quotas

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Maximum number of characters per document</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Maximum size per document</td>
<td>20 MB</td>
</tr>
<tr>
<td>Maximum size of translatable text in a single document</td>
<td>1 MB</td>
</tr>
<tr>
<td>Maximum number of target languages in a batch job request</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of documents in batch</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Maximum size of total documents in batch</td>
<td>5 GB</td>
</tr>
<tr>
<td>Maximum number of concurrent batch translation jobs</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of queued batch translation jobs</td>
<td>1000</td>
</tr>
<tr>
<td>Transactions per second for the StartTextTranslationJob API action</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second for the DescribeTextTranslationJob API action</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second for the ListTextTranslationJobs API action</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second for the StopTextTranslationJob API action</td>
<td>5</td>
</tr>
</tbody>
</table>

### Custom terminology quotas

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum custom terminology file size</td>
<td>10 MB</td>
</tr>
<tr>
<td>Maximum number of custom terminology files per AWS account per AWS Region</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of target languages per custom terminology file</td>
<td>10</td>
</tr>
<tr>
<td>Maximum source and target text length per custom terminology term</td>
<td>200 bytes</td>
</tr>
<tr>
<td>Maximum number of terminology files per TranslateText or StartTextTranslationJob request</td>
<td>1</td>
</tr>
</tbody>
</table>
### Service quotas

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions per second for the ImportTerminology API action</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second for the GetTerminology API action</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second for the ListTerminologies API action</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second for the DeleteTerminology API action</td>
<td>5</td>
</tr>
</tbody>
</table>

### Parallel data quotas

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of parallel data resources per AWS account per AWS Region</td>
<td>1000</td>
</tr>
<tr>
<td>Maximum parallel data input file size</td>
<td>5 GB</td>
</tr>
<tr>
<td>Maximum number of source languages in a parallel data resource</td>
<td>1</td>
</tr>
<tr>
<td>Maximum size of a single segment or record in a parallel data input file</td>
<td>1000 bytes</td>
</tr>
<tr>
<td>Maximum number of concurrent create or update operations for parallel data resources</td>
<td>1</td>
</tr>
<tr>
<td>Transactions per second for the CreateParallelData API action</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second for the GetParallelData API action</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second for the ListParallelData API action</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second for the UpdateParallelData API action</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second for the DeleteParallelData API action</td>
<td>5</td>
</tr>
</tbody>
</table>
# Document history for Amazon Translate

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

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New feature: Brevity</strong></td>
<td>Translate now supports brevity for real-time text translations. Brevity reduces the length of the translation output for most translations (compared to the translation output without brevity). For more information, see <a href="#">Using brevity in Amazon Translate</a>.</td>
<td>October 31, 2023</td>
</tr>
<tr>
<td>Language auto-detection for document input to real-time translations</td>
<td>You can now use language auto-detection when you input a document to real-time translations (console or API). For more information, see <a href="#">Real-time translations</a>.</td>
<td>August 3, 2023</td>
</tr>
<tr>
<td>Word (.docx) files as input to real-time translations</td>
<td>You can now use .docx files (in addition to text files and HTML files) as input to real-time translations (console or API). For more information, see <a href="#">Real-time translations</a>.</td>
<td>July 17, 2023</td>
</tr>
<tr>
<td>Enhancements to custom terminology</td>
<td>Translate now supports enhancements to the custom terminology feature that improve translation fluency and accuracy. For more information, see <a href="#">Customizing your translations with custom terminology</a>.</td>
<td>June 30, 2023</td>
</tr>
<tr>
<td>Text or HTML files as input to real-time translations</td>
<td>You can now use text files or HTML files as input to real-time translations (console or API). For more information, see <a href="#">Real-time translations</a>.</td>
<td>May 23, 2023</td>
</tr>
<tr>
<td>New action allowed in the TranslateReadOnly policy</td>
<td>Amazon Translate now allows the TranslateDocument action in the TranslateReadOnly managed policy. For more information, see <a href="#">AWS managed policy: TranslateReadOnly</a>.</td>
<td>May 23, 2023</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td><strong>Translate now supports additional regions</strong></td>
<td>Translate now supports additional regions for asynchronous batch processing. For more information, see <a href="#">Asynchronous batch processing with Amazon Translate</a>.</td>
<td>March 28, 2023</td>
</tr>
<tr>
<td><strong>Increased input size for real-time</strong></td>
<td>You can now input up to 10,000 characters for real-time translations. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/userguide/gs-console.html">Getting started in Amazon Translate (console)</a>.</td>
<td>December 16, 2022</td>
</tr>
<tr>
<td><strong>Support for nested input folders</strong></td>
<td>You can now provide nested input folders to batch translation jobs. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/dg/batch-features.html">Running a batch translation job</a> in Amazon Translate.</td>
<td>November 18, 2022</td>
</tr>
<tr>
<td><strong>Support for auto-language detection</strong></td>
<td>You can now auto-detect the source language in batch translation jobs. As a result, you can now input documents with different source languages in batch translation jobs. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/dg/batch-features.html">Running a batch translation job</a> in Amazon Translate.</td>
<td>November 18, 2022</td>
</tr>
<tr>
<td><strong>Support for multiple target languages</strong></td>
<td>You can now specify multiple target languages in batch translation jobs. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/dg/batch-features.html">Running a batch translation job</a> in Amazon Translate.</td>
<td>October 10, 2022</td>
</tr>
<tr>
<td><strong>Support for tags</strong></td>
<td>You can now tag <a href="https://docs.aws.amazon.com/translate/latest/dg/parallel-data-features.html">ParallelData</a> and <a href="https://docs.aws.amazon.com/translate/latest/dg/terminology-features.html">Custom Terminology</a> resources in Amazon Translate. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/dg/parallel-data-features.html">Tagging your resources in Amazon Translate</a>.</td>
<td>October 6, 2022</td>
</tr>
<tr>
<td><strong>Formality support for additional languages</strong></td>
<td>You can now set the translation formality level for Dutch, Korean, and Mexican Spanish in Amazon Translate. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/dg/formality-features.html">Setting formality in Amazon Translate</a>.</td>
<td>October 5, 2022</td>
</tr>
<tr>
<td><strong>Separate API Reference</strong></td>
<td>The Amazon Translate API Reference is now a separate document from the Developer Guide. For more information, see <a href="https://docs.aws.amazon.com/translate/latest/dg/api.html">Amazon Translate API Reference</a>.</td>
<td>August 25, 2022</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>You can now set the formality level for your translation output. For more information, see <a href="#">Setting formality in Amazon Translate</a>.</td>
<td>February 22, 2022</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>You can now mask profane words and phrases in your translation output. For more information, see <a href="#">Masking profane words and phrases in Amazon Translate</a>.</td>
<td>November 24, 2021</td>
</tr>
<tr>
<td><strong>AWS PrivateLink support</strong></td>
<td>You can now establish a private connection between your VPC and Amazon Translate by using AWS PrivateLink. For more information, see <a href="#">Amazon Translate and interface VPC endpoints (AWS PrivateLink)</a>.</td>
<td>November 24, 2021</td>
</tr>
<tr>
<td><strong>Parallel data update</strong></td>
<td>You can now create parallel data resources that use any of the languages that are supported by Amazon Translate. You no longer need to use English as one of the languages. For more information about parallel data, see <a href="#">Customizing your translations with parallel data (Active Custom Translation)</a>.</td>
<td>November 15, 2021</td>
</tr>
<tr>
<td><strong>Custom terminology directionality</strong></td>
<td>You can now create multi-directional terminology, in which any language can be the source language or a target language. For more information, see <a href="#">Creating a custom terminology</a>.</td>
<td>November 11, 2021</td>
</tr>
<tr>
<td><strong>New languages</strong></td>
<td>Amazon Translate now supports the following languages: Irish, Marathi, Portuguese (Portugal), and Punjabi. For all of the languages that Amazon Translate supports, see <a href="#">Supported languages and language codes</a>.</td>
<td>November 10, 2021</td>
</tr>
<tr>
<td><strong>New custom encryption settings</strong></td>
<td>You can now encrypt your translation output by using your own customer managed key that you manage in AWS Key Management Service. For more information, see <a href="#">Running a batch translation job</a>.</td>
<td>November 5, 2021</td>
</tr>
<tr>
<td><strong>New file format support</strong></td>
<td>Amazon Translate now supports XML Localization Interchange File Format (XLIFF) files for asynchronous batch processing. For all supported formats, see <a href="#">Supported file formats</a>.</td>
<td>June 9, 2021</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>EventBridge integration</strong></td>
<td>Amazon Translate now sends events to Amazon EventBridge to notify you about changes that affect your translation jobs and parallel data resources. For more information, see <a href="#">Monitoring Amazon Translate events with Amazon EventBridge</a>.</td>
<td>June 4, 2021</td>
</tr>
<tr>
<td><strong>New quota</strong></td>
<td>Amazon Translate now supports up to 1000 queued batch translation jobs. For all Amazon Translate quotas, see <a href="#">Guidelines and limits</a>.</td>
<td>April 23, 2021</td>
</tr>
<tr>
<td><strong>Quota increase</strong></td>
<td>The maximum size for a parallel data input file has increased from 1 MB to 5 MB. For all Amazon Translate quotas, see <a href="#">Guidelines and limits</a>.</td>
<td>March 31, 2021</td>
</tr>
<tr>
<td><strong>New languages</strong></td>
<td>Amazon Translate now supports the following languages: Armenian, Catalan, Farsi (Persian), Filipino Tagalog, Gujarati, Haitian Creole, Icelandic, Kannada, Kazakh, Lithuanian, Macedonian, Malayalam, Maltese, Mongolian, Sinhala, Telugu, Uzbek, and Welsh. For all of the languages that Amazon Translate supports, see <a href="#">Supported languages and language codes</a>.</td>
<td>November 23, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>You can now customize batch translation jobs by using parallel data, which consists of examples of source text and their translations. Jobs that use parallel data are called Active Custom Translation jobs. During these jobs, Amazon Translate adapts the translation output to reflect the examples in the parallel data. For more information, see <a href="#">Customizing your translations with parallel data (Active Custom Translation)</a>.</td>
<td>November 23, 2020</td>
</tr>
<tr>
<td>New file format support</td>
<td>Amazon Translate now supports the following Office Open XML file formats as input for asynchronous batch processing: Word document (.docx), PowerPoint presentation (.pptx), Excel workbook (.xlsx). For more information, see Starting a batch translation job.</td>
<td>July 29, 2020</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>New language</td>
<td>Amazon Translate now supports the Spanish (Mexico) language for translation. For all supported languages, see Supported languages and language codes.</td>
<td>April 30, 2020</td>
</tr>
<tr>
<td>New region</td>
<td>Amazon Translate supports asynchronous batch processing in the Europe (London) Region. For all of the AWS regions where asynchronous batch processing is available, see Region availability.</td>
<td>April 20, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Translate adds asynchronous batch translation functionality. For more information, see Asynchronous batch processing.</td>
<td>December 23, 2019</td>
</tr>
<tr>
<td>New regions</td>
<td>Amazon Translate adds support for the Asia Pacific (Hong Kong), Asia Pacific (Sydney), EU (London), EU (Paris), EU (Stockholm), and US West (N. California) Regions. For a complete list of the AWS Regions supported by Amazon Translate, see the AWS Region Table or AWS Regions and Endpoints in the Amazon Web Services General Reference.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>New languages</td>
<td>Amazon Translate adds new language for translation: Afrikaans, Albanian, Amharic, Azerbaijani, Bengali, Bosnian, Bulgarian, Canadian-French, Croatian, Dari, Estonian, Georgian, Hausa, Latvian, Pashto, Serbian, Slovak, Slovenian, Somali, Swahili, Tagalog, and Tamil. For a list of the language combinations that Amazon Translate can translate directly, see Supported languages.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td><strong>New languages</strong></td>
<td>Amazon Translate adds new languages for translation: Greek, Hungarian, Romanian, Thai, Ukrainian, Urdu, and Vietnamese. For a list of the language combinations that Amazon Translate can translate directly, see [Supported languages].</td>
<td>October 3, 2019</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Translate adds <a href="#">FedRAMP compliance</a>. For more information, see [Compliance].</td>
<td>July 31, 2019</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Translate adds <a href="#">SOC compliance</a>. For more information, see [Compliance].</td>
<td>May 30, 2019</td>
</tr>
<tr>
<td><strong>New regions</strong></td>
<td>Amazon Translate adds support for the Asia Pacific (Mumbai), Asia Pacific (Singapore), Asia Pacific (Tokyo), and Canada (Central) Regions. For a complete list of the AWS Regions supported by Amazon Translate, see the <a href="#">AWS Region Table</a> or <a href="#">AWS Regions and Endpoints</a> in the <a href="#">Amazon Web Services General Reference</a>.</td>
<td>May 8, 2019</td>
</tr>
<tr>
<td><strong>New languages</strong></td>
<td>Amazon Translate adds new languages for translation: Hindi, Malay, Norwegian, and Persian. For a list of the language combinations that Amazon Translate can translate directly, see [Supported languages].</td>
<td>May 6, 2019</td>
</tr>
<tr>
<td><strong>New region</strong></td>
<td>Amazon Translate adds support for the EU (Frankfurt) and Asia Pacific (Seoul) Regions. For a complete list of the AWS Regions supported by Amazon Translate, see the <a href="#">AWS Region Table</a> or <a href="#">AWS Regions and Endpoints</a> in the <a href="#">Amazon Web Services General Reference</a>.</td>
<td>February 28, 2019</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Translate adds <a href="#">PCI compliance</a>. For more information, see [Compliance].</td>
<td>December 12, 2018</td>
</tr>
<tr>
<td>New feature</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td>Amazon Translate adds four new APIs and the custom terminology feature to</td>
<td>November 27, 2018</td>
<td></td>
</tr>
<tr>
<td>give you more control over your translation. By using a custom terminology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>with your translation requests, you can make sure that your brand names,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>character names, model names, and other unique content is translated exactly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>the way you want it, every time, regardless of the standard translation or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context. For more information, see Custom terminology.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New languages</td>
<td>November 20, 2018</td>
<td></td>
</tr>
<tr>
<td>Amazon Translate now translates documents in the following languages:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Danish, Dutch, Finnish, Hebrew, Indonesian, Korean, Polish, and Swedish.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon Translate continues to improve direct translation by significantly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reducing the number of unsupported language pairs. For the language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>combinations that Amazon Translate can translate directly, see Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>languages.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New feature</td>
<td>October 29, 2018</td>
<td></td>
</tr>
<tr>
<td>Amazon Translate adds direct translation between supported languages other</td>
<td></td>
<td></td>
</tr>
<tr>
<td>than English. For the language combinations that Amazon Translate can</td>
<td></td>
<td></td>
</tr>
<tr>
<td>translate directly, see Supported languages.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New feature</td>
<td>October 25, 2018</td>
<td></td>
</tr>
<tr>
<td>Amazon Translate adds HIPAA compliance. For more information, see Compliance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New feature</td>
<td>July 17, 2018</td>
<td></td>
</tr>
<tr>
<td>Amazon Translate adds multiple new languages for translation: Chinese</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Tradition), Czech, Italian, Japanese, Russian, and Turkish. For a list of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>languages that Amazon Translate supports, see Supported languages.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New feature</td>
<td>April 4, 2018</td>
<td></td>
</tr>
<tr>
<td>Amazon Translate adds support for automatic source language detection. For</td>
<td></td>
<td></td>
</tr>
<tr>
<td>more information, see How Amazon Translate works.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New guide (p. 121)  
This is the first release of the *Amazon Translate Developer Guide*.

November 29, 2017
API reference

The Amazon Translate API Reference is now a separate document. For more information, see Amazon Translate API Reference.
AWS Glossary

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