aws

Developer Guide

Amazon Translate

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Translate: Developer Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

What is Amazon Translate? 1
Use cases 1
First-time user 2
Amazon Translate pricing 2
Amazon Translate API Reference 2
Supported languages 3
Supported languages 3
Languages supported by Amazon Translate features 7
How it works 8
Supported formats for the input content 8
Customizing your translations 8
Automatic language detection 9
Exception handling 9
Setting up 10
Sign up for an AWS account 10
Create a user with administrative access 10
Set up the AWS CLI 12
Grant programmatic access 12
Working with AWS SDKs 14
Getting started 16
Getting started (console) 16
Getting started (AWS CLI) 18
Translate text using the command line 18
Next step 19
Getting started (SDK) 19
Using the SDK for Java 19
Using the AWS SDK for Python 20
Other SDK examples 20
Translation processing modes 21
Real-time translation 21
Real-time translation (console) 21
Real-time translation (API) 26
Asynchronous batch processing 29
Region availability 29

Prerequisites 30
Running a job 35
Monitoring and analyzing 42
Getting results 44
Customizing your translations 47
Using do-not-translate tags 47
Using do-not-translate with the console 48
Using do-not-translate with the API 48
Customizing with custom terminology 49
Creating a custom terminology 50
Using custom terminologies 52
Example using SDK for Python 53
Encrypting your terminology 55
Best practices 55
Using brevity 57
Using the brevity setting 57
Supported languages 57
Masking profanity 58
Using the profanity setting 59
Unsupported languages 59
Setting formality 60
Using the formality setting 60
Supported languages 61
Customizing with parallel data 62
Region availability 63
Parallel data input files for Amazon Translate 63
Adding parallel data 69
Viewing and managing parallel data 73
Code examples 77
Basics 77
Actions 78
Scenarios 96
Build an Amazon Transcribe streaming app 96
Building an Amazon Lex chatbot 97
Building an Amazon SNS application 98
Create an application to analyze customer feedback 100

Get started with translate jobs ... 106
Tagging ... 109
Tagging a new resource ... 110
Viewing, editing, and deleting tags ... 111
Security ... 113
Data protection ... 114
Encryption at rest ... 115
Encryption in transit ... 116
Identity and Access Management ... 116
Audience ... 116
Authenticating with identities ... 117
Managing access using policies ... 120
How Amazon Translate works with IAM ... 123
Identity-based policy examples ... 130
AWS managed policies ... 136
Troubleshooting ... 139
Monitoring ... 140
Monitoring with CloudWatch ... 143
Logging Amazon Translate API calls with AWS CloudTrail ... 145
CloudWatch metrics and dimensions for Amazon Translate ... 147
Monitoring with EventBridge ... 149
Compliance validation ... 152
Resilience ... 153
Infrastructure security ... 153
VPC endpoints (AWS PrivateLink) ... 154
Considerations for Amazon Translate VPC endpoints ... 154
Creating an interface VPC endpoint for Amazon Translate ... 154
Creating a VPC endpoint policy for Amazon Translate ... 155
Guidelines and quotas ... 157
Supported AWS Regions ... 157
Compliance ... 157
Throttling ... 157
Guidelines ... 157
Service quotas ... 158
Document history ... 161
API reference ... 174

AWS Glossary 175

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 for information about the languages that Amazon Translate supports.

Topics

- Use cases
- Are you a first-time user of Amazon Translate?
- Amazon Translate pricing
- Amazon Translate API Reference

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 – Introduces Amazon Translate.
2. Getting started with Amazon Translate – Explains how to set up your AWS account and start using Amazon Translate.
3. Code examples for Amazon Translate using AWS SDKs – Use the code examples to explore the Amazon Translate APIs.

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.

Topics

- Supported languages
- Languages supported by Amazon Translate features

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`.

Language	Language code
Afrikaans	af
Albanian	sq
Amharic	am
Arabic	ar
Armenian	hy

Language	Language code
Azerbaijani	az
Bengali	bn
Bosnian	bs
Bulgarian	bg
Catalan	ca
Chinese (Simplified)	zh
Chinese (Traditional)	zh-TW
Croatian	hr
Czech	cs
Danish	da
Dari	fa-AF
Dutch	nl
English	en
Estonian	et
Farsi (Persian)	fa
Filipino, Tagalog	tl
Finnish	fi
French	fr
French (Canada)	fr-CA
Georgian	ka

Language	Language code
German	de
Greek	el
Gujarati	gu
Haitian Creole	ht
Hausa	ha
Hebrew	he
Hindi	hi
Hungarian	hu
Icelandic	is
Indonesian	id
Irish	ga
Italian	it
Japanese	ja
Kannada	kn
Kazakh	kk
Korean	ko
Latvian	lv
Lithuanian	lt
Macedonian	mk
Malay	ms

Language	Language code
Malayalam	ml
Maltese	mt
Marathi	mr
Mongolian	mn
Norwegian (Bokmål)	no
Pashto	ps
Polish	pl
Portuguese (Brazil)	pt
Portuguese (Portugal)	pt-PT
Punjabi	pa
Romanian	ro
Russian	ru
Serbian	sr
Sinhala	si
Slovak	sk
Slovenian	sl
Somali	so
Spanish	es
Spanish (Mexico)	es-MX
Swahili	sw

Language	Language code
Swedish	sv
Tamil	ta
Telugu	te
Thai	th
Turkish	tr
Ukrainian	uk
Urdu	ur
Uzbek	uz
Vietnamese	vi
Welsh	cy

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.
- Brevity – For the languages supported by this feature, see Using brevity in Amazon Translate.
- Profanity masking – For the languages supported by this feature, see Masking profane words and phrases in Amazon Translate.
- Formality – For the languages supported by this feature, see Setting formality in Amazon Translate.

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.

Topics

- Supported formats for the input content
- Customizing your translations
- Automatic language detection
- Exception handling

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.
- **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
- Create a user with administrative access
- Install and configure the AWS Command Line Interface (AWS CLI)
- Grant programmatic access
- Using this service with an AWS SDK

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

1. Open https://portal.aws.amazon.com/billing/signup.
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 a 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 a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and 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 a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- 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*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center 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.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM

Which user needs programmatic access?	To	By
		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*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

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)
- Getting started (AWS CLI)
- Getting started (SDK)

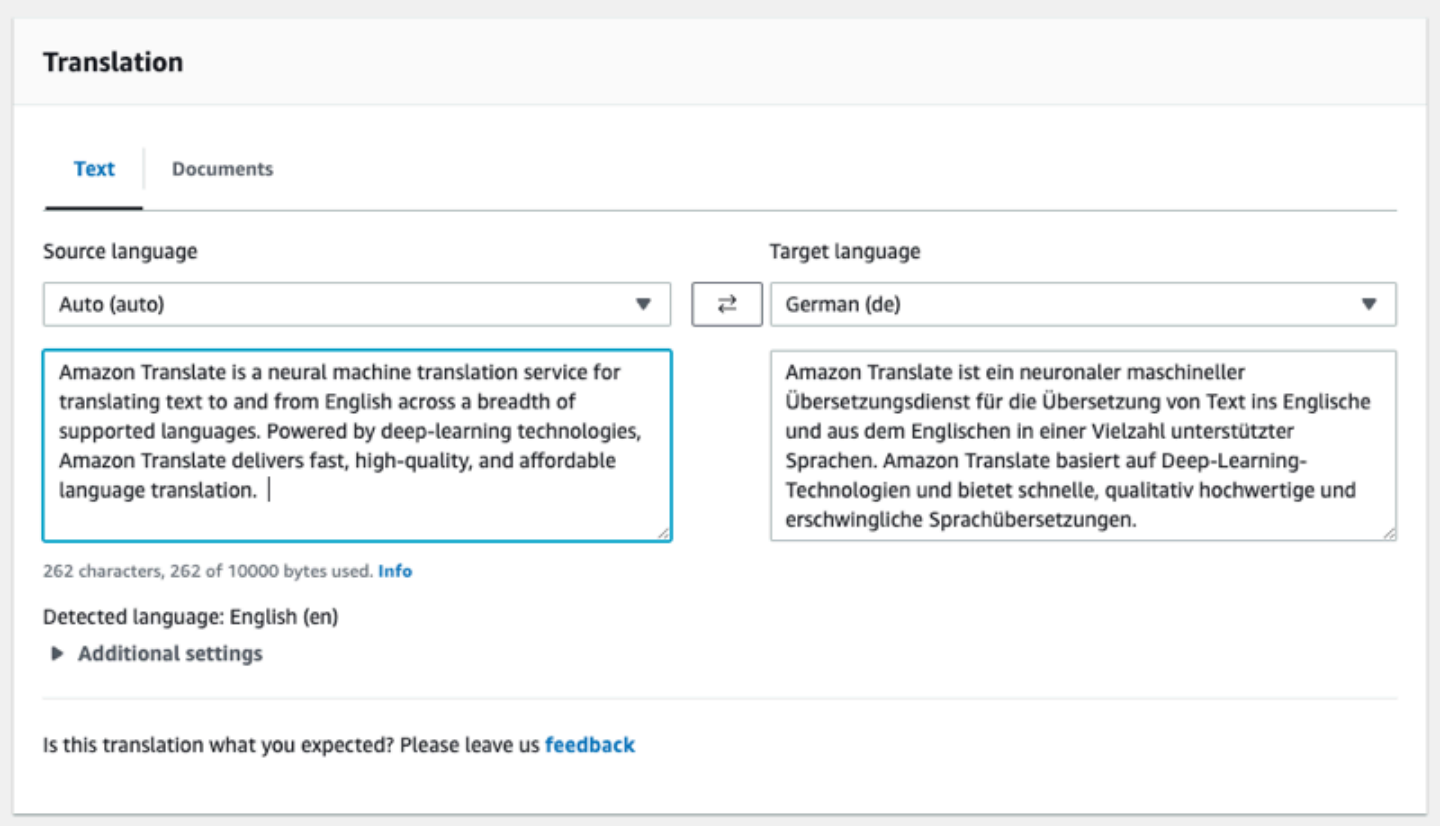
Getting started (console)

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, we recommend that you do so before proceeding.

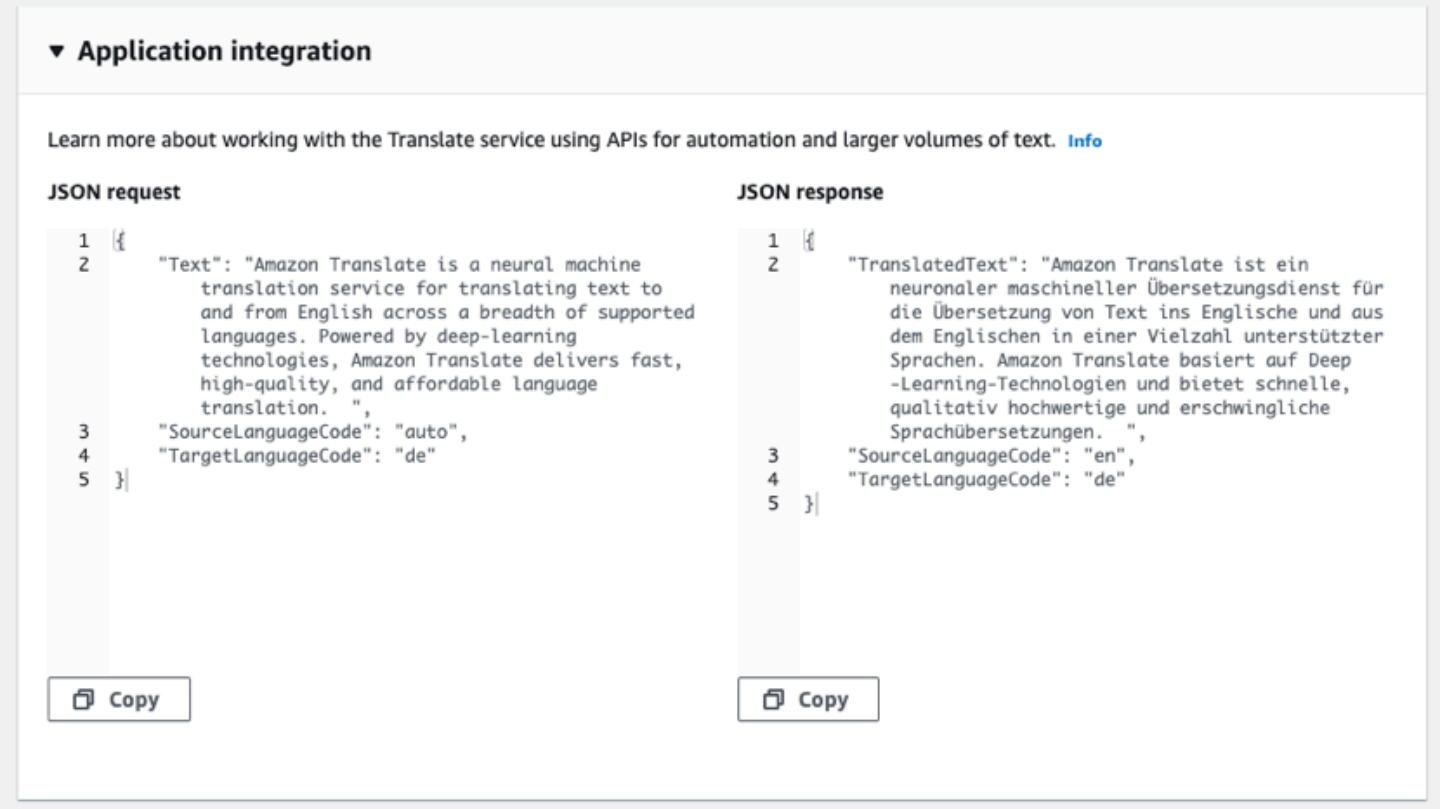
Open the Amazon Translate console.

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).

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.

```
aws translate translate-text \
            --region region \
```

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

The response is the following JSON:

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

Next step

To see other ways to use Amazon Translate see Code examples for Amazon Translate using AWS SDKs.

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 and Python. 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
- Translating text using the AWS SDK for Python (Boto)
- Other SDK examples

Translating text using the AWS SDK for Java

AWS provides a GitHub example of how to use 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 2.x.

Translating text using the AWS SDK for Python (Boto)

The following example shows how to use the TranslateText operation in Python. To run the example, install the Python SDK via the AWS CLI. For instructions, see the section called "Set up the AWS CLI".

```
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

Other SDK examples

See Code examples for Amazon Translate using AWS SDKs for examples that use .NET and SAP ABAP.

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 – 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 – 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
- Real-time translation using the API

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.

Topics

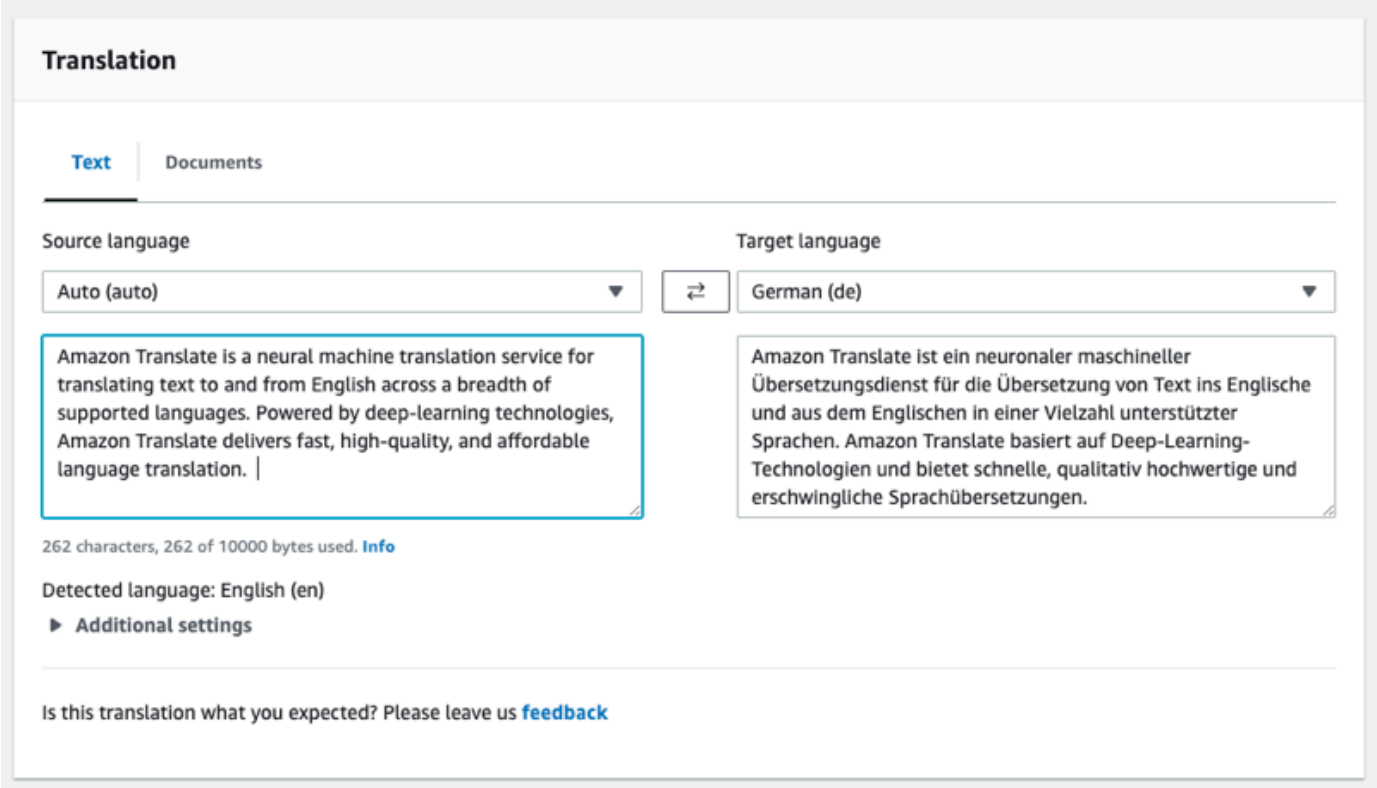
- Translate text
- Translate a document
- View equivalent API request and response data

- Use translation features

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.

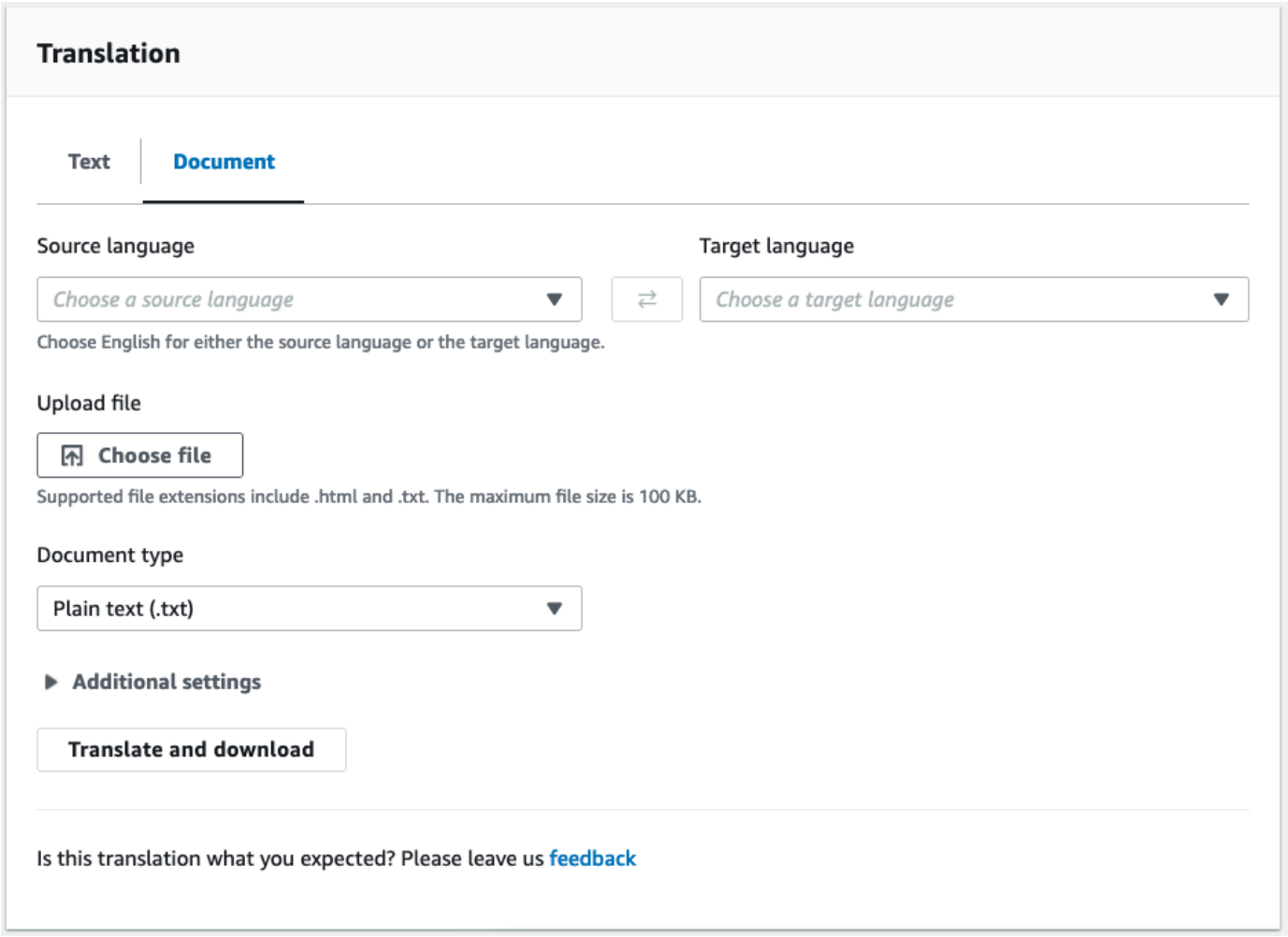


Translate a document

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.



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**.

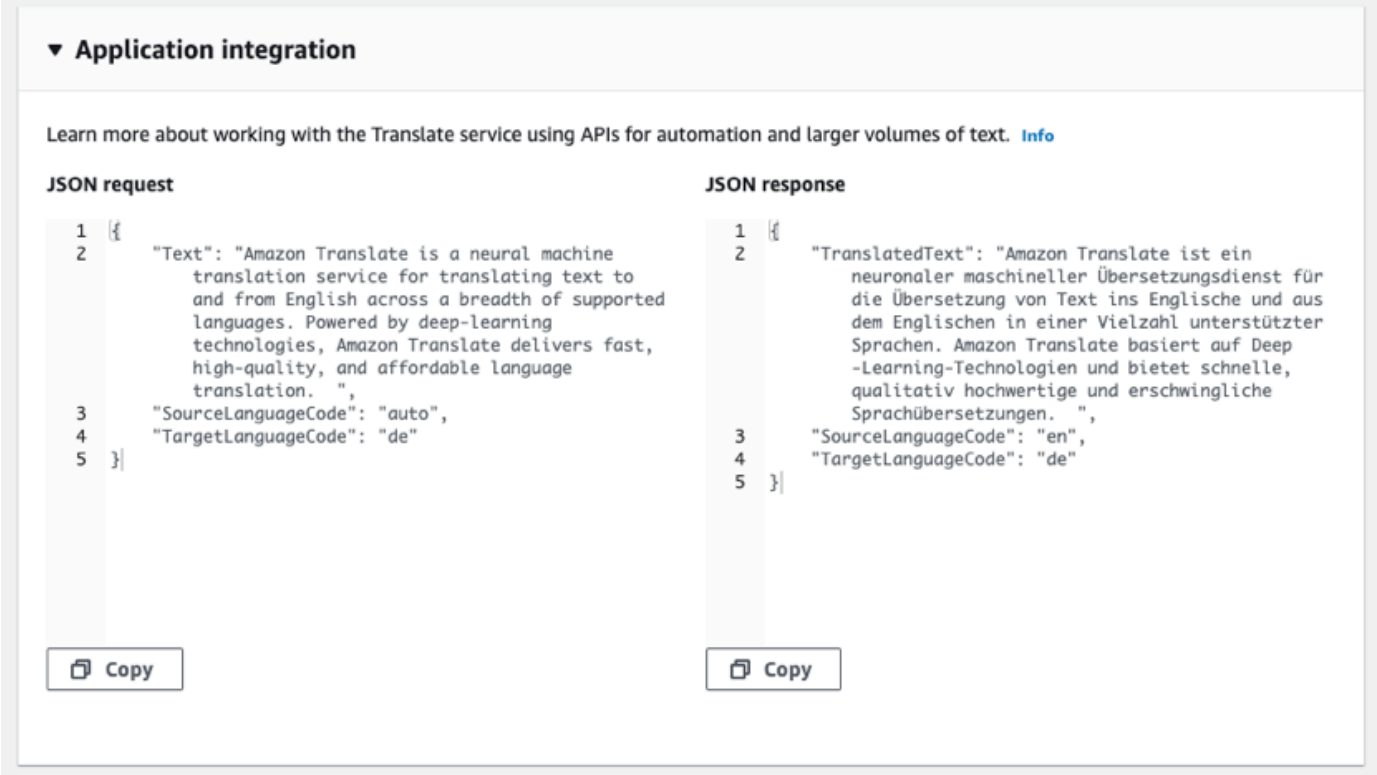
After the translation task completes, choose **Download translation** to download the translated document to your local hard drive. The format of the translated document (text, HTML, or Word) matches the input document.

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.

   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.

   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.

   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.
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.

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`:

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

2. In the AWS CLI, run the following command:

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

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

```
{
    "TargetLanguageCode": "fr",
    "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 ContentType=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
- Running a batch translation job
- Monitoring and analyzing batch translation jobs
- Getting batch translation results

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:

- 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 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"
      ],
      "Resource": "arn:aws:s3:::output-bucket-name/*"
    }
  ]
}
```

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":
  {
    "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

```
{
  "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

```
{
  "Effect": "Allow",
  "Principal":
  {
    "AWS":
    [
      "arn:aws:iam::444455556666:role/AmazonTranslateServiceRoleS3FullAccess",
      "arn:aws:iam::444455556666:admin"
    ]
  },
  "Action":
  [
    "kms:Decrypt",
    "kms:CreateGrant",
    "kms:GenerateDataKey",
    "kms:RetireGrant",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

For more information, see Allowing users in other accounts to use a AWS KMS key in the *AWS Key Management Service Developer Guide*

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

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.

```
$ aws translate start-text-translation-job \
> --input-data-config ContentType=application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet,S3Uri=s3://amzn-s3-demo-bucket/input/ \
> --output-data-config S3Uri=s3://amzn-s3-demo-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:

```
{
    "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.

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. 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.

`--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.

`--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).

To check the status of your translation job, use the `describe-text-translation-job` command.

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 4446f95f20c88a4b347449d3671fbe3d
```

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

```
{
    "TextTranslationJobProperties": {
        "JobId": "4446f95f20c88a4b347449d3671fbe3d",
        "JobName": "my-translation-job",
        "JobStatus": "COMPLETED",
        "JobDetails": {
            "TranslatedDocumentsCount": 0,
            "DocumentsWithErrorsCount": 0,
            "InputDocumentsCount": 1
        },
```

```
        "SourceLanguageCode": "en",
        "TargetLanguageCodes": [
            "es",
            "it"
        ],
        "SubmittedTime": 1598661012.468,
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket/input/",
            "ContentType": "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket/output/111122223333-
TranslateText-4446f95f20c88a4b347449d3671fbe3d/"
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/my-iam-role"
    }
}
```

If the `JobStatus` value is `IN_PROGRESS`, allow a few minutes to pass, and run `describe-text-translation-job` again until the status is `COMPLETED`. When the job completes, you can download the translation results at the location provided by the `S3Uri` field under `OutputDataConfig`.

Amazon Translate API

To submit a batch translation job by using the Amazon Translate API, use the StartTextTranslationJob operation.

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 check's a job's status by using the AWS CLI to run the DescribeTextTranslationJob command:

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

This command returns the following output:

```
{
  "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",
    "TargetLanguageCodes": [
      "fr"
    ],
    "JobName": "batch-test",
    "SubmittedTime": 1576544017.357,
    "JobStatus": "COMPLETED",
    "Message": "Your job has completed successfully.",
    "JobDetails": {
      "InputDocumentsCount": 77,
      "DocumentsWithErrorsCount": 0,
      "TranslatedDocumentsCount": 77
    },
    "OutputDataConfig": {
      "S3Uri": "s3://bucket-name/output/012345678901-
TranslateText-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:

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

This command returns the following output:

```
{
  "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.


```
{
  "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
- Customizing your translations with custom terminology
- Using brevity in Amazon Translate
- Masking profane words and phrases in Amazon Translate
- Setting formality in Amazon Translate
- Customizing your translations with parallel data (Active Custom Translation)

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
- Using do-not-translate with the API

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:

```
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:

```
{
```

```
    "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. 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.

Topics

- Creating a custom terminology
- Using custom terminologies
- Custom Terminology example using the AWS SDK for Python (Boto)
- Encrypting your terminology
- Best practices

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

```
        <seg>Amazon Photos</seg>
      </tuv>
      <tuv xml:lang="es">
        <seg>Amazon Photos</seg>
      </tuv>
    </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 (^).

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

This uses the selected Custom Terminology to translate this text as "Avez-vous déjà fait des achats avec Amazon Famille?" instead of the direct (but undesirable) translation "Avez-vous déjà fait des achats avec Famille Amazon?"

The following example shows how to use the same terminology file in Python.

```
import boto3

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

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

```
print("Translated text: " + response.get('TranslatedText'))
print("\n")
```

For more information on using the Amazon Translate operations with Custom Terminologies, see API Operations.

Custom Terminology example using the AWS SDK for Python (Boto)

The following example shows how to use the Custom Terminology operations in Python. To run the example, install the Python SDK via the AWS CLI. For instructions, see the section called "Set up the AWS CLI".

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

    import boto3

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

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

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

    file_data = bytearray(data)

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

    print("Getting the imported terminology...")
    response = translate.get_terminology(Name='my-first-terminology',
TerminologyDataFormat='CSV')
    print("Received terminology: "),
    print(response.get('TerminologyProperties'))
```

```
    print("The terminology data file can be downloaded here: " +
 response.get('TerminologyDataLocation').get('Location'))
    print("\n")

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

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

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

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

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

    file_data = bytearray(data)

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

    print("Translating 'Prime Video' from English to French with no terminology...")
```

```
    response = translate.translate_text(Text="Prime Video", SourceLanguageCode="en",
 TargetLanguageCode="fr")
    print("Translated text: " + response.get('TranslatedText'))
    print("\n")

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

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

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
- Supported languages

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 to one of the languages in the following table.
- From any of the languages in the following table to English.

Language	Language code
French	fr
German	de
Italian	it
Portuguese (Brazil)	pt
Spanish	es

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
- Unsupported languages

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.

Unsupported languages

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

Language	Language code
Bengali	bn
Hindi	hi
Malayalam	ml
Punjabi	pa
Sinhala	si
Vietnamese	vi

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

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
- Supported languages

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.

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.

Language	Language code
Dutch	nl
French	fr
French (Canada)	fr-CA
German	de
Hindi	hi
Italian	it
Japanese	ja
Korean	ko
Portuguese (Portugal)	pt-PT
Spanish	es
Spanish (Mexico)	es-MX

For all the languages that Amazon Translate supports, see Supported languages and language codes.

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":

```
Bonjour, comment allez-vous ?
Comment allez-vous ?
Salut, comment allez-vous ?
Comment allez-vous ?
```

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
- Adding your parallel data to Amazon Translate
- Viewing and managing your parallel data in Amazon Translate

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.

Example parallel data

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

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 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

```
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.
```

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.
- 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:

```
{
  "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 \
> --parallel-data-config S3Uri=s3://input-bucket/parallel-data-file.tsv,Format=TSV
```

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.

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.

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",
            "Arn": "arn:aws:translate:us-west-2:111122223333:parallel-data/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
        }
    ]
}
```

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:

```
$ 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",
        "Arn": "arn:aws:translate:us-west-2:111122223333:parallel-data/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:

```
$ aws translate update-parallel-data \
> --name my-parallel-data \
> --parallel-data-config S3Uri=s3://input-bucket/parallel-data-file.tsv,Format=TSV
```

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.

Code examples for Amazon Translate using AWS SDKs

The following code examples show how to use Amazon Translate with an AWS software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Basic examples for Amazon Translate using AWS SDKs
  - Actions for Amazon Translate using AWS SDKs
    - Use DescribeTextTranslationJob with an AWS SDK
    - Use ListTextTranslationJobs with an AWS SDK
    - Use StartTextTranslationJob with an AWS SDK
    - Use StopTextTranslationJob with an AWS SDK
    - Use TranslateText with an AWS SDK or CLI
- Scenarios for Amazon Translate using AWS SDKs
  - Build an Amazon Transcribe streaming app
  - Create an Amazon Lex chatbot to engage your website visitors
  - Build a publish and subscription application that translates messages
  - Create an application that analyzes customer feedback and synthesizes audio
  - Get started with Amazon Translate jobs using an AWS SDK

Basic examples for Amazon Translate using AWS SDKs

The following code examples show how to use the basics of Amazon Translate with AWS SDKs.

Examples

- Actions for Amazon Translate using AWS SDKs
  - Use DescribeTextTranslationJob with an AWS SDK
  - Use ListTextTranslationJobs with an AWS SDK
  - Use StartTextTranslationJob with an AWS SDK
  - Use StopTextTranslationJob with an AWS SDK
  - Use TranslateText with an AWS SDK or CLI

Actions for Amazon Translate using AWS SDKs

The following code examples demonstrate how to perform individual Amazon Translate actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the Amazon Translate API and are code excerpts from larger programs that must be run in context. You can see actions in context in Scenarios for Amazon Translate using AWS SDKs .

The following examples include only the most commonly used actions. For a complete list, see the Amazon Translate API Reference.

Examples

- Use DescribeTextTranslationJob with an AWS SDK
- Use ListTextTranslationJobs with an AWS SDK
- Use StartTextTranslationJob with an AWS SDK
- Use StopTextTranslationJob with an AWS SDK
- Use TranslateText with an AWS SDK or CLI

Use `DescribeTextTranslationJob` with an AWS SDK

The following code examples show how to use `DescribeTextTranslationJob`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get started with translate jobs

.NET

AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// The following example shows how to retrieve the details of
    /// a text translation job using Amazon Translate.
    /// </summary>
    public class DescribeTextTranslation
    {
        public static async Task Main()
        {
            var client = new AmazonTranslateClient();

            // The Job Id is generated when the text translation job is started
            // with a call to the StartTextTranslationJob method.
            var jobId = "1234567890abcdef01234567890abcde";

            var request = new DescribeTextTranslationJobRequest
            {
                JobId = jobId,
            };

            var jobProperties = await DescribeTranslationJobAsync(client,
request);

            DisplayTranslationJobDetails(jobProperties);
        }

        /// <summary>
        /// Retrieve information about an Amazon Translate text translation job.
```

```
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client
 object.</param>
        /// <param name="request">The DescribeTextTranslationJobRequest object.</
param>
        /// <returns>The TextTranslationJobProperties object containing
        /// information about the text translation job..</returns>
        public static async Task<TextTranslationJobProperties>
 DescribeTranslationJobAsync(
            AmazonTranslateClient client,
            DescribeTextTranslationJobRequest request)
        {
            var response = await client.DescribeTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                return response.TextTranslationJobProperties;
            }
            else
            {
                return null;
            }
        }

        /// <summary>
        /// Displays the properties of the text translation job.
        /// </summary>
        /// <param name="jobProperties">The properties of the text translation
        /// job returned by the call to DescribeTextTranslationJobAsync.</param>
        public static void
 DisplayTranslationJobDetails(TextTranslationJobProperties jobProperties)
        {
            if (jobProperties is null)
            {
                Console.WriteLine("No text translation job properties found.");
                return;
            }

            // Display the details of the text translation job.
            Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
        }
    }
```

- For API details, see DescribeTextTranslationJob in *AWS SDK for .NET API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   "Gets the properties associated with an asynchronous batch translation job."
   "Includes properties such as name, ID, status, source and target languages,
and input/output Amazon Simple Storage Service (Amazon S3) buckets."
   TRY.
       oo_result = lo_xl8->describetexttranslationjob(      "oo_result is
returned for testing purposes."
         EXPORTING
           iv_jobid        = iv_jobid
         ).
       MESSAGE 'Job description retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex .
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex .
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- For API details, see DescribeTextTranslationJob in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListTextTranslationJobs` with an AWS SDK

The following code examples show how to use `ListTextTranslationJobs`.

.NET

AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// List Amazon Translate translation jobs, along with details about each
job.
    /// </summary>
    public class ListTranslationJobs
    {
        public static async Task Main()
        {
            var client = new AmazonTranslateClient();
            var filter = new TextTranslationJobFilter
            {
                JobStatus = "COMPLETED",
            };

            var request = new ListTextTranslationJobsRequest
            {
                MaxResults = 10,
                Filter = filter,
            };

            await ListJobsAsync(client, request);
        }
```

```
        /// <summary>
        /// List Amazon Translate text translation jobs.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client
object.</param>
        /// <param name="request">An Amazon Translate
        /// ListTextTranslationJobsRequest object detailing which text
        /// translation jobs are of interest.</param>
        public static async Task ListJobsAsync(
            AmazonTranslateClient client,
            ListTextTranslationJobsRequest request)
        {
            ListTextTranslationJobsResponse response;

            do
            {
                response = await client.ListTextTranslationJobsAsync(request);

ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

                request.NextToken = response.NextToken;
            }
            while (response.NextToken is not null);
        }

        /// <summary>
        /// List existing translation job details.
        /// </summary>
        /// <param name="properties">A list of Amazon Translate text
        /// translation jobs.</param>
        public static void
ShowTranslationJobDetails(List<TextTranslationJobProperties> properties)
        {
            properties.ForEach(prop =>
            {
                Console.WriteLine($"{prop.JobId}: {prop.JobName}");
                Console.WriteLine($"Status: {prop.JobStatus}");
                Console.WriteLine($"Submitted time: {prop.SubmittedTime}");
            });
        }
    }
```

- For API details, see ListTextTranslationJobs in *AWS SDK for .NET API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   "Gets a list of the batch translation jobs that you have submitted."

   DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.

   "Create an ABAP object for filtering using jobname."
   CREATE OBJECT lo_filter
     EXPORTING
       iv_jobname = iv_jobname.

   TRY.
       oo_result = lo_xl8->listtexttranslationjobs(      "oo_result is returned
for testing purposes."
         EXPORTING
           io_filter        = lo_filter
         ).
       MESSAGE 'Jobs retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex .
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8invalidfilterex .
       MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex .
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- For API details, see ListTextTranslationJobs in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartTextTranslationJob` with an AWS SDK

The following code examples show how to use `StartTextTranslationJob`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get started with translate jobs

.NET

AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// This example shows how to use Amazon Translate to process the files in
    /// an Amazon Simple Storage Service (Amazon S3) bucket. The translated
results
    /// will also be stored in an Amazon S3 bucket.
    /// </summary>
    public class BatchTranslate
```

```
    {
        public static async Task Main()
        {
            var contentType = "text/plain";

            // Set this variable to an S3 bucket location with a folder."
            // Input files must be in a folder and not at the bucket root."
            var s3InputUri = "s3://amzn-s3-demo-bucket1/FOLDER/";
            var s3OutputUri = "s3://amzn-s3-demo-bucket2/";

            // This role must have permissions to read the source bucket and to
 read and
            // write to the destination bucket where the translated text will be
 stored.
            var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/
S3TranslateRole";

            var client = new AmazonTranslateClient();

            var inputConfig = new InputDataConfig
            {
                ContentType = contentType,
                S3Uri = s3InputUri,
            };

            var outputConfig = new OutputDataConfig
            {
                S3Uri = s3OutputUri,
            };

            var request = new StartTextTranslationJobRequest
            {
                JobName = "ExampleTranslationJob",
                DataAccessRoleArn = dataAccessRoleArn,
                InputDataConfig = inputConfig,
                OutputDataConfig = outputConfig,
                SourceLanguageCode = "en",
                TargetLanguageCodes = new List<string> { "fr" },
            };

            var response = await StartTextTranslationAsync(client, request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
```

```
                Console.WriteLine($"{response.JobId}: {response.JobStatus}");
            }
        }

        /// <summary>
        /// Start the Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized AmazonTranslateClient object.</
param>
        /// <param name="request">The request object that includes details such
        /// as source and destination bucket names and the IAM Role that will
        /// be used to access the buckets.</param>
        /// <returns>The StartTextTranslationResponse object that includes the
        /// details of the request response.</returns>
        public static async Task<StartTextTranslationJobResponse>
 StartTextTranslationAsync(AmazonTranslateClient client,
 StartTextTranslationJobRequest request)
        {
            var response = await client.StartTextTranslationJobAsync(request);
            return response;
        }
    }
```

- For API details, see StartTextTranslationJob in *AWS SDK for .NET API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    "Starts an asynchronous batch translation job."
    "Use batch translation jobs to translate large volumes of text across
 multiple documents at once."

    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
```

```
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    CREATE OBJECT lo_inputdataconfig
      EXPORTING
        iv_s3uri       = iv_input_data_s3uri
        iv_contenttype = iv_input_data_contenttype.

    "Create an ABAP object for the output data config."
    CREATE OBJECT lo_outputdataconfig
      EXPORTING
        iv_s3uri = iv_output_data_s3uri.

    "Create an internal table for target languages."
    CREATE OBJECT lo_targetlanguagecodes
      EXPORTING
        iv_value = iv_targetlanguagecode.
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        oo_result = lo_xl8->starttexttranslationjob(      "oo_result is returned
 for testing purposes."
          EXPORTING
            io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode
          ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8invparamvalueex .
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
```

```
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00 .
        MESSAGE 'Amazon Translate does not support translation from the language
of the source text into the requested target language.' TYPE 'E'.
    ENDTRY.
```

- For API details, see StartTextTranslationJob in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StopTextTranslationJob` with an AWS SDK

The following code examples show how to use `StopTextTranslationJob`.

.NET

AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Shows how to stop a running Amazon Translation Service text translation
/// job.
/// </summary>
public class StopTextTranslationJob
{
    public static async Task Main()
    {
```

```
            var client = new AmazonTranslateClient();
            var jobId = "1234567890abcdef01234567890abcde";

            var request = new StopTextTranslationJobRequest
            {
                JobId = jobId,
            };

            await StopTranslationJobAsync(client, request);
        }

        /// <summary>
        /// Sends a request to stop a text translation job.
        /// </summary>
        /// <param name="client">Initialized AmazonTrnslateClient object.</param>
        /// <param name="request">The request object to be passed to the
        /// StopTextJobAsync method.</param>
        public static async Task StopTranslationJobAsync(
            AmazonTranslateClient client,
            StopTextTranslationJobRequest request)
        {
            var response = await client.StopTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId} as status:
{response.JobStatus}");
            }
        }
    }
```

- For API details, see StopTextTranslationJob in *AWS SDK for .NET API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   "Stops an asynchronous batch translation job that is in progress."

   TRY.
       oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned
for testing purposes."
         EXPORTING
           iv_jobid        = iv_jobid
         ).
       MESSAGE 'Translation job stopped.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex .
       MESSAGE 'An internal server error occurred.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex .
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- For API details, see StopTextTranslationJob in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `TranslateText` with an AWS SDK or CLI

The following code examples show how to use `TranslateText`.

.NET

AWS SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   using System;
   using System.IO;
```

```
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// Take text from a file stored a Amazon Simple Storage Service (Amazon S3)
    /// object and translate it using the Amazon Transfer Service.
    /// </summary>
    public class TranslateText
    {
        public static async Task Main()
        {
            // If the region you want to use is different from the region
            // defined for the default user, supply it as a parameter to the
            // Amazon Translate client object constructor.
            var client = new AmazonTranslateClient();

            // Set the source language to "auto" to request Amazon Translate to
            // automatically detect te language of the source text.

            // You can get a list of the languages supposed by Amazon Translate
            // in the Amazon Translate Developer's Guide here:
            //      https://docs.aws.amazon.com/translate/latest/dg/what-is.html
            string srcLang = "en"; // English.
            string destLang = "fr"; // French.

            // The Amazon Simple Storage Service (Amazon S3) bucket where the
            // source text file is stored.
            string srcBucket = "amzn-s3-demo-bucket";
            string srcTextFile = "source.txt";

            var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
            var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

            ShowText(srcText, destText);
        }

        /// <summary>
        /// Use the Amazon S3 TransferUtility to retrieve the text to translate
        /// from an object in an S3 bucket.
        /// </summary>
```

```
        /// <param name="srcBucket">The name of the S3 bucket where the
        /// text is stored.
        /// </param>
        /// <param name="srcTextFile">The key of the S3 object that
        /// contains the text to translate.</param>
        /// <returns>A string representing the source text.</returns>
        public static async Task<string> GetSourceTextAsync(string srcBucket,
 string srcTextFile)
        {
            string srcText = string.Empty;

            var s3Client = new AmazonS3Client();
            TransferUtility utility = new TransferUtility(s3Client);

            using var stream = await utility.OpenStreamAsync(srcBucket,
 srcTextFile);

            StreamReader file = new System.IO.StreamReader(stream);

            srcText = file.ReadToEnd();
            return srcText;
        }

        /// <summary>
        /// Use the Amazon Translate Service to translate the document from the
        /// source language to the specified destination language.
        /// </summary>
        /// <param name="client">The Amazon Translate Service client used to
        /// perform the translation.</param>
        /// <param name="srcLang">The language of the source text.</param>
        /// <param name="destLang">The destination language for the translated
        /// text.</param>
        /// <param name="text">A string representing the text to ranslate.</
param>
        /// <returns>The text that has been translated to the destination
        /// language.</returns>
        public static async Task<string>
 TranslatingTextAsync(AmazonTranslateClient client, string srcLang, string
 destLang, string text)
        {
            var request = new TranslateTextRequest
            {
                SourceLanguageCode = srcLang,
                TargetLanguageCode = destLang,
```

```
                Text = text,
            };

            var response = await client.TranslateTextAsync(request);

            return response.TranslatedText;
        }

        /// <summary>
        /// Show the original text followed by the translated text.
        /// </summary>
        /// <param name="srcText">The original text to be translated.</param>
        /// <param name="destText">The translated text.</param>
        public static void ShowText(string srcText, string destText)
        {
            Console.WriteLine("Source text:");
            Console.WriteLine(srcText);
            Console.WriteLine();
            Console.WriteLine("Translated text:");
            Console.WriteLine(destText);
        }
    }
```

- For API details, see TranslateText in *AWS SDK for .NET API Reference*.

PowerShell

Tools for PowerShell

Example 1: Converts the specified English text to French. The text to convert can also be passed as the -Text parameter.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- For API details, see TranslateText in *AWS Tools for PowerShell Cmdlet Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   "Translates input text from the source language to the target language."
   TRY.
       oo_result = lo_xl8->translatetext(      "oo_result is returned for
testing purposes."
         EXPORTING
           iv_text        = iv_text
           iv_sourcelanguagecode = iv_sourcelanguagecode
           iv_targetlanguagecode = iv_targetlanguagecode
         ).
       MESSAGE 'Translation completed.' TYPE 'I'.
     CATCH /aws1/cx_xl8detectedlanguage00 .
       MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
     CATCH /aws1/cx_xl8internalserverex .
       MESSAGE 'An internal server error occurred.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex .
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex .
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8serviceunavailex .
       MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE
'E'.
     CATCH /aws1/cx_xl8textsizelmtexcdex .
       MESSAGE 'The size of the text you submitted exceeds the size limit. '
TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex .
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
     CATCH /aws1/cx_xl8unsuppedlanguage00 .
       MESSAGE 'Amazon Translate does not support translation from the language
of the source text into the requested target language. ' TYPE 'E'.
   ENDTRY.
```

- For API details, see TranslateText in *AWS SDK for SAP ABAP API reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Scenarios for Amazon Translate using AWS SDKs

The following code examples show you how to implement common scenarios in Amazon Translate with AWS SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon Translate or combined with other AWS services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

- Build an Amazon Transcribe streaming app
- Create an Amazon Lex chatbot to engage your website visitors
- Build a publish and subscription application that translates messages
- Create an application that analyzes customer feedback and synthesizes audio
- Get started with Amazon Translate jobs using an AWS SDK

Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

JavaScript

SDK for JavaScript (v3)

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Create an Amazon Lex chatbot to engage your website visitors

The following code examples show how to create a chatbot to engage your website visitors.

Java

SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK for JavaScript (v3)

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example Building an Amazon Lex chatbot in the AWS SDK for JavaScript developer guide.

Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Build a publish and subscription application that translates messages

The following code examples show how to create an application that has subscription and publish functionality and translates messages.

.NET

AWS SDK for .NET

Shows how to use the Amazon Simple Notification Service .NET API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

Java

SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

Kotlin

SDK for Kotlin

Shows how to use the Amazon SNS Kotlin API to create an application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to create a web app, see the full example on GitHub.

For complete source code and instructions on how to create a native Android app, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Create an application that analyzes customer feedback and synthesizes audio

The following code examples show how to create an application that analyzes customer comment cards, translates them from their original language, determines their sentiment, and generates an audio file from the translated text.

.NET

AWS SDK for .NET

This example application analyzes and stores customer feedback cards. Specifically, it fulfills the need of a fictitious hotel in New York City. The hotel receives feedback from guests in various languages in the form of physical comment cards. That feedback is uploaded into the app through a web client. After an image of a comment card is uploaded, the following steps occur:

- Text is extracted from the image using Amazon Textract.
- Amazon Comprehend determines the sentiment of the extracted text and its language.
- The extracted text is translated to English using Amazon Translate.
- Amazon Polly synthesizes an audio file from the extracted text.

The full app can be deployed with the AWS CDK. For source code and deployment instructions, see the project in GitHub.

Services used in this example

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK for Java 2.x

This example application analyzes and stores customer feedback cards. Specifically, it fulfills the need of a fictitious hotel in New York City. The hotel receives feedback from guests in

various languages in the form of physical comment cards. That feedback is uploaded into the app through a web client. After an image of a comment card is uploaded, the following steps occur:

- Text is extracted from the image using Amazon Textract.
- Amazon Comprehend determines the sentiment of the extracted text and its language.
- The extracted text is translated to English using Amazon Translate.
- Amazon Polly synthesizes an audio file from the extracted text.

The full app can be deployed with the AWS CDK. For source code and deployment instructions, see the project in GitHub.

Services used in this example

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK for JavaScript (v3)

This example application analyzes and stores customer feedback cards. Specifically, it fulfills the need of a fictitious hotel in New York City. The hotel receives feedback from guests in various languages in the form of physical comment cards. That feedback is uploaded into the app through a web client. After an image of a comment card is uploaded, the following steps occur:

- Text is extracted from the image using Amazon Textract.
- Amazon Comprehend determines the sentiment of the extracted text and its language.
- The extracted text is translated to English using Amazon Translate.
- Amazon Polly synthesizes an audio file from the extracted text.

The full app can be deployed with the AWS CDK. For source code and deployment instructions, see the project in GitHub. The following excerpts show how the AWS SDK for JavaScript is used inside of Lambda functions.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";
```

```
/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
```

```
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});
```

```
  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Services used in this example

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK for Ruby

This example application analyzes and stores customer feedback cards. Specifically, it fulfills the need of a fictitious hotel in New York City. The hotel receives feedback from guests in various languages in the form of physical comment cards. That feedback is uploaded into the app through a web client. After an image of a comment card is uploaded, the following steps occur:

- Text is extracted from the image using Amazon Textract.
- Amazon Comprehend determines the sentiment of the extracted text and its language.
- The extracted text is translated to English using Amazon Translate.
- Amazon Polly synthesizes an audio file from the extracted text.

The full app can be deployed with the AWS CDK. For source code and deployment instructions, see the project in GitHub.

Services used in this example

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Get started with Amazon Translate jobs using an AWS SDK

The following code example shows how to:

- Start an asynchronous batch translation job.
- Wait for the asynchronous job to complete.
- Describe the asynchronous job.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    CREATE OBJECT lo_inputdataconfig
      EXPORTING
        iv_s3uri       = iv_input_data_s3uri
```

```
        iv_contenttype = iv_input_data_contenttype.

    "Create an ABAP object for the output data config."
    CREATE OBJECT lo_outputdataconfig
      EXPORTING
        iv_s3uri = iv_output_data_s3uri.

    "Create an internal table for target languages."
    CREATE OBJECT lo_targetlanguagecodes
      EXPORTING
        iv_value = iv_targetlanguagecode.
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
          EXPORTING
            io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode
          ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
      CATCH /aws1/cx_xl8invparamvalueex .
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time. '
TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00 .
        MESSAGE 'Amazon Translate does not support translation from the language
of the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

    "Get the job ID."
    DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).
```

```
    "Wait for translate job to complete."
    DATA(lo_des_translation_result) = lo_xl8-
>describetexttranslationjob( iv_jobid = lv_jobid ).
    WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
 lv_jobid ).
    ENDWHILE.

    TRY.
        oo_result = lo_xl8->describetexttranslationjob(      "oo_result is
 returned for testing purposes."
          EXPORTING
            iv_jobid        = lv_jobid
          ).
        MESSAGE 'Job description retrieved.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
 TYPE 'E'.
    ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - DescribeTextTranslationJob
  - StartTextTranslationJob

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

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

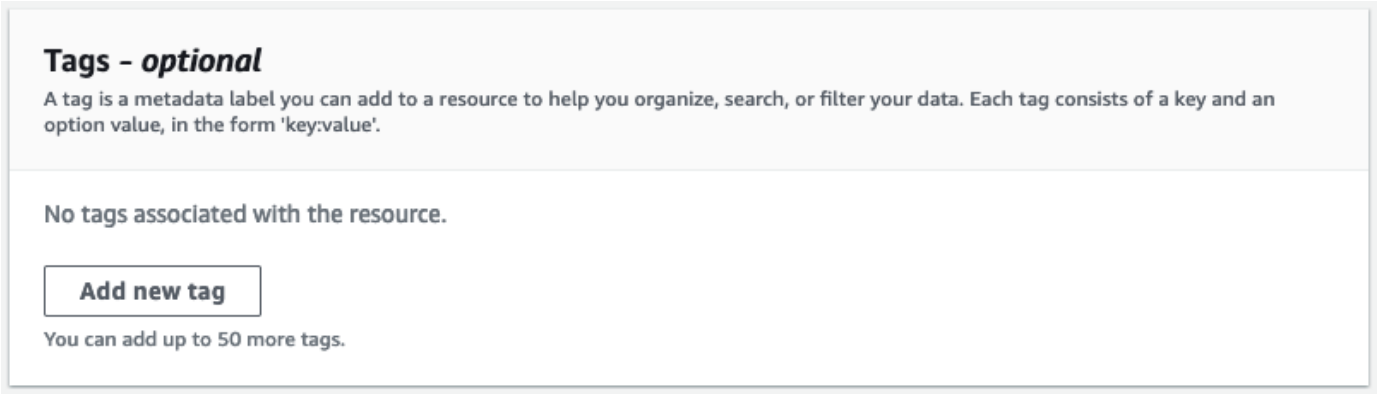
- Tagging a new resource
- Viewing, updating, and deleting tags associated with a resource

Tagging a new resource

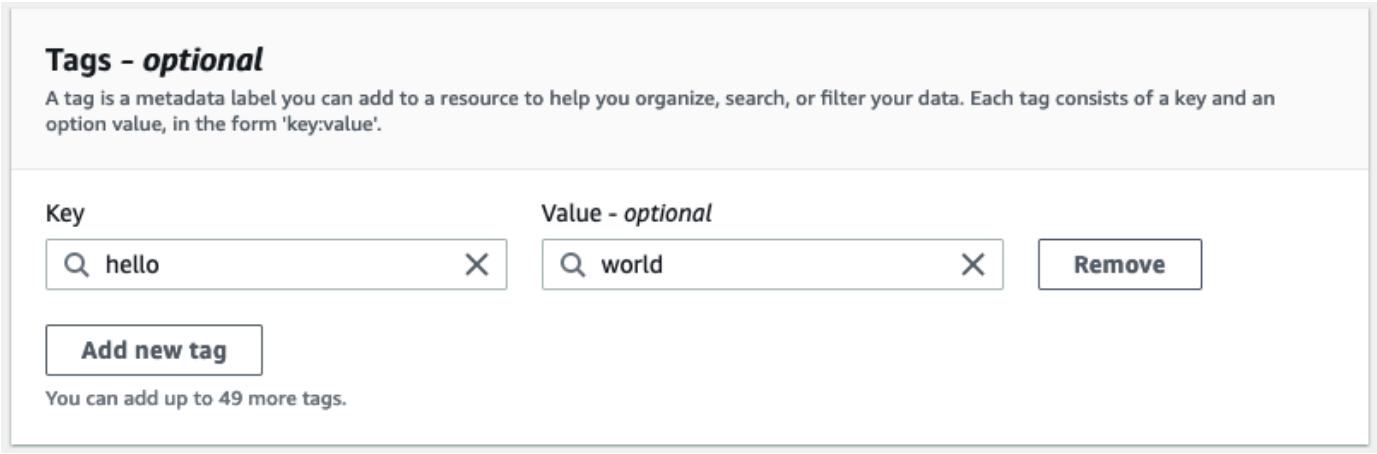
You can add tags to a **ParallelData** or **Custom Terminology** resource when you create it.

To add tags to a new resource (console)

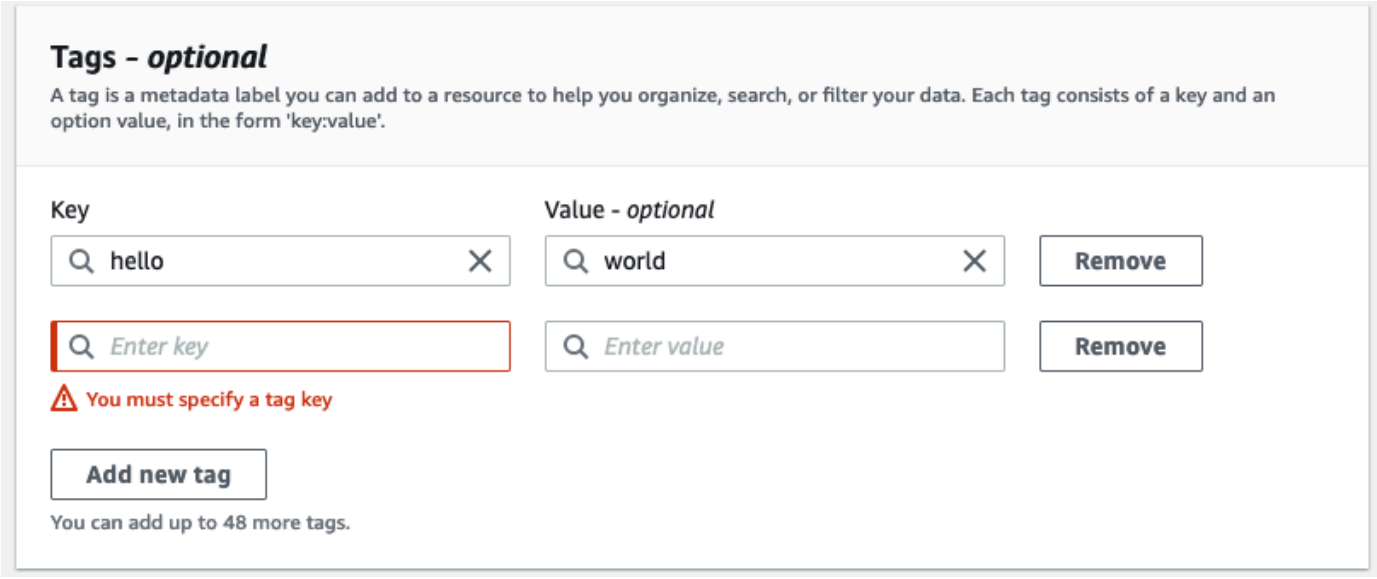
1. Sign in to the Amazon Translate console.
2. From the left navigation pane, select the resource (`Parallel data` or `Custom terminology`) that you want to create.
3. Chose **Create parallel data** or **Create terminology**. The console displays the main 'create' page for your resource. At the end of 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.

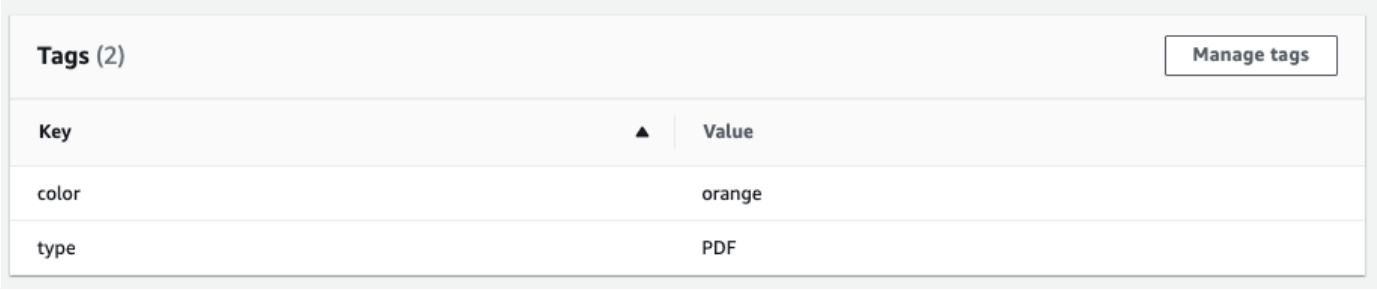
```
aws translate create-parallel-data \
--name "myTest" \
--parallel-data-config "{\"format\": \"CSV\",  \
            "S3Uri\": \"s3://test-input/TEST.csv\"}" \
--tags "[{\"Key\": \"color\",\"Value\": \"orange\"}]"
```

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.



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
- Identity and Access Management for Amazon Translate
- Monitoring Amazon Translate
- Compliance validation for Amazon Translate
- Resilience in Amazon Translate
- Infrastructure security in Amazon Translate
- Amazon Translate and interface VPC endpoints (AWS PrivateLink)

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
- Encryption in transit

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.

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.

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
- Authenticating with identities
- Managing access using policies
- How Amazon Translate works with IAM
- Identity-based policy examples for Amazon Translate
- AWS managed policies for Amazon Translate
- Troubleshooting Amazon Translate identity and access

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.

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.

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.

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 AWS Signature Version 4 for 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 AWS Multi-factor authentication in IAM 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 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 Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role 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 Create a role for a third-party identity provider (federation) 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 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 Cross account resource access in IAM 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.

- **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
- **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 Create a role to delegate permissions to an AWS service 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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances 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 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 Define custom IAM permissions with customer managed policies 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 Choose between managed policies and inline policies 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 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 Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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

IAM feature	Amazon Translate support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how Amazon Translate and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

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 Define custom IAM permissions with customer managed 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.

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 Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Amazon Translate

Supports policy actions: 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 `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.

Policy resources for Amazon Translate

Supports policy resources: 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 `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.

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.

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 Define permissions with ABAC authorization 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.

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 Switch from a user to an IAM 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.

Forward access sessions for Amazon Translate

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for Amazon Translate

Supports service roles: Yes

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 Create 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.

Service-linked roles for Amazon Translate

Supports service-linked roles: No

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
- Allow access to the Amazon Translate console
- Allow users to view their own permissions
- Specify resources in a policy
- Permissions for using customer managed keys with custom terminologies

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 or AWS managed policies for job functions 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 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 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 Validate policies with IAM Access Analyzer 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 Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM 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.

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

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:GetRole",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
             ],
            "Resource": "*"

        }
    ]
}
```

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.

```
{
    "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:

  arn:*partition*:translate:*region*:*account*:terminology/*terminology-name*/LATEST

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

  arn:*partition*:translate:*region*:*account*:parallel-data/*parallel-data-name*

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

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "translate:*",
```

```
        "Resource": [
             "arn:aws:translate:us-west-2:123456789012:terminology/*"
        ]
}
```

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

Example

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "translate:GetParallelData",
        "Resource": [
             "arn:aws:translate:us-west-2:123456789012:parallel-data/test-parallel-
data"
        ]
}
```

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.

```
{
    "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",
                "kms:GenerateDataKey",
```

```
                "kms:GetKeyPolicy",
                "kms:PutKeyPolicy",
                "kms:RetireGrant"
            ],
            "Resource": "*"
        }
    ]
}
```

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 `DeleteTermionlogy` 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

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

Topics

- AWS managed policy: TranslateFullAccess
- AWS managed policy: TranslateReadOnly
- Amazon Translate updates to AWS managed policies

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.

```
{
    "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.

```
{
```

```
  "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.

Change	Description	Date
TranslateReadOnly – Update to an existing policy	Amazon Translate now allows the `TranslateDocument` action in the Translate ReadOnly policy	May 23, 2023
Amazon Translate started tracking changes	Amazon Translate started tracking changes for its AWS managed policies.	May 23, 2023

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
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon Translate resources

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.
- 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 Cross account resource access in IAM in the *IAM User Guide*.

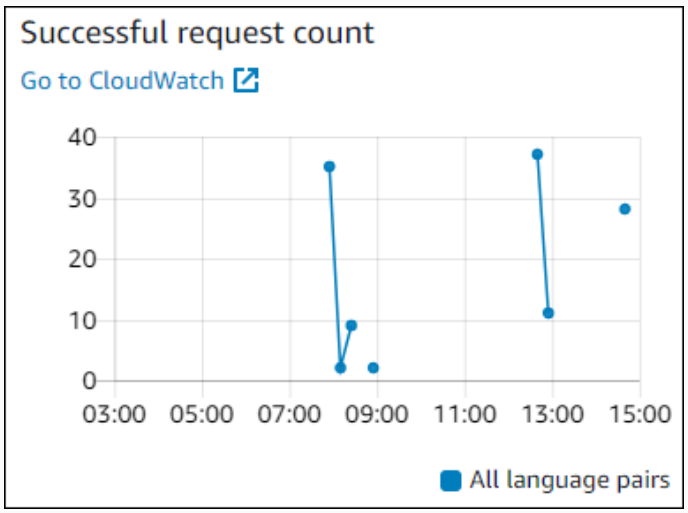
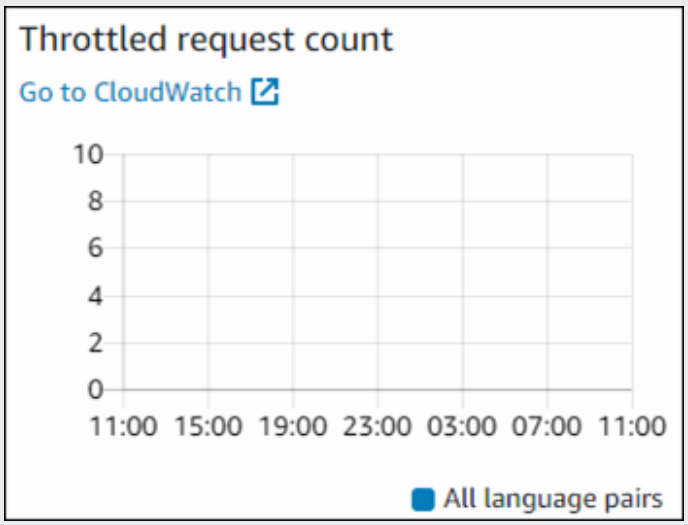
Monitoring Amazon Translate

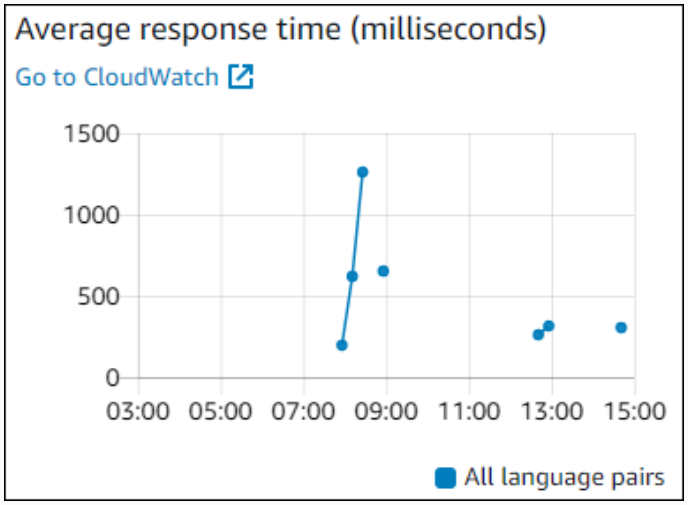
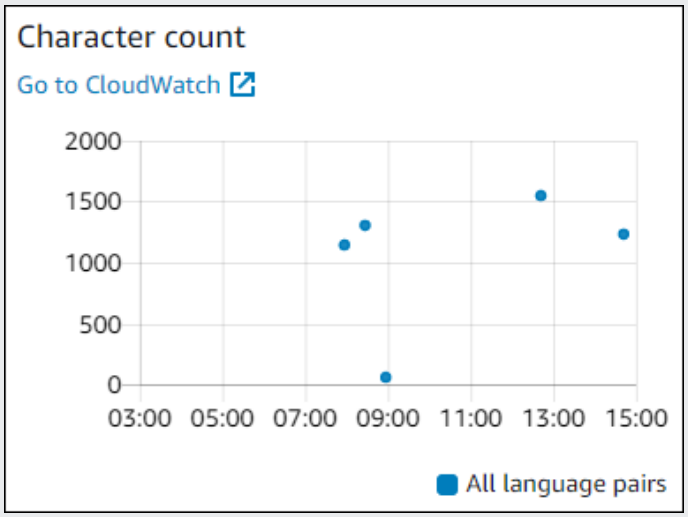
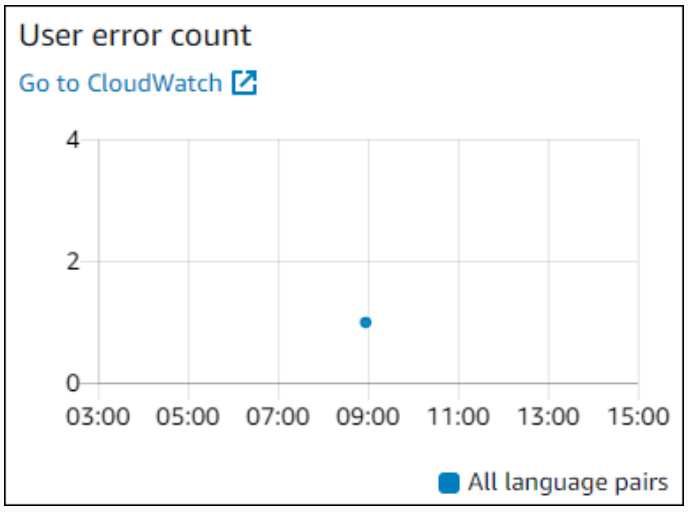
Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Translate and your solutions. AWS provides various tools that you can use to monitor Amazon Translate. You can configure some of these tools to monitor your solutions for you. We recommend that you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

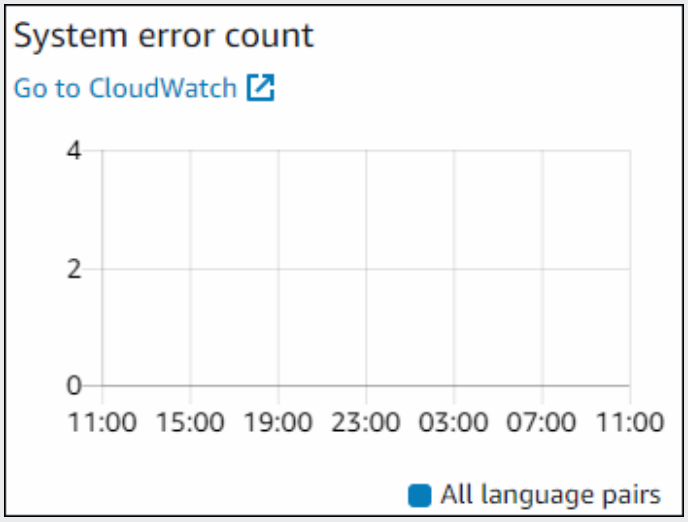
Amazon Translate provides preconfigured graphs that show you the most important metrics for your solution. Each graph offers a window into your solution's performance. To get different views of how your solution is performing over time, you can change the time range that the graphs show.

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.

Graph	Description
	Successful request count The number of successful requests made to Amazon Translate during the specified time period.
	Throttled request count 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.

Graph	Description
	Average response time The average length of time that it took Amazon Translate to process your request during the specified time period.
	Character count 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.
	User error count The number of user errors that occurred during the specified time period. User errors are in the HTTP error code range 400-499.

Graph	Description
	System error count The number of system errors that occurred during the specified time period. System errors are in the HTTP error code range 500-599.

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.

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

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

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

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

Viewing Amazon Translate metrics

View Amazon Translate metrics in the CloudWatch console.

To view metrics (CloudWatch console)

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Metrics**, choose **All Metrics**, and then choose **AWS/Translate**.
3. Choose the dimension, choose a metric name, and choose **Add to graph**.

4. Choose a value for the date range. The metric count for the specified date range is displayed in the graph.

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
- Understanding Amazon Translate log file entries

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 tegions 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.metal1.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": "f56da956-284e-4983-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

Metric	Description
`CharacterCount`	The number of billable characters in requests. Valid dimensions: Language pair, Operation Valid statistics: Average, Maximum, Minimum, Sum Unit: Count
`ResponseTime`	The time that it took to respond to a request. Valid dimensions: Language pair, Operation

Metric	Description
	Valid statistics: Data samples, Average Unit: For Data samples, count. For Average statistics, milliseconds.
`ServerErrorCount`	The number of server errors. The HTTP response code range for a server error is 500 to 599. Valid dimension: Operation Valid statistics: Average, Sum Unit: Count
`SuccessfulRequestCount`	The number of successful translation requests. The response code for a successful request is 200 to 299. Valid dimension: Operation Valid statistics: Average, Sum Unit: Count
`ThrottledCount`	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*. Valid dimension: Operation Valid statistics: Average, Sum Unit: Count

Metric	Description
UserErrorCount	The number of user errors that occurred. The HTTP response code range for a user error is 400 to 499. Valid dimension: Operation Valid statistics: Average, Sum Unit: Count

CloudWatch Dimensions for Amazon Translate

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

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

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.

```
{
 "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.

```
{
 "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": "STATUS"
 }
}
```

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.

```
{
 "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": "STATUS",
    "latestUpdateAttemptStatus": "STATUS",
    "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.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

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.

```
{
            "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.

```
{
            "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
- Compliance
- Throttling
- Guidelines
- Service quotas

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

Description	Limit
Character encoding	UTF-8
Maximum input text	10,000 bytes
Maximum number of characters per document	100,000
Maximum document size	100,000 bytes

Asynchronous batch translation quotas

Description	Limit
Character encoding	UTF-8
Maximum number of characters per document	1,000,000
Maximum size per document	20 MB
Maximum size of translatable text in a single document	1 MB
Maximum number of target languages in a batch job request	10
Maximum number of documents in batch	1,000,000
Maximum size of total documents in batch	5 GB
Maximum number of concurrent batch translation jobs	10
Maximum number of queued batch translation jobs	1000

Description	Limit
Transactions per second for the StartTextTranslationJob API action	5
Transactions per second for the DescribeTextTranslationJob API action	10
Transactions per second for the ListTextTranslationJobs API action	10
Transactions per second for the StopTextTranslationJob API action	5

Custom terminology quotas

Description	Limit
Maximum custom terminology file size	10 MB
Maximum number of custom terminology files per AWS account per AWS Region	100
Maximum number of target languages per custom terminology file	10
Maximum source and target text length per custom terminology term	200 bytes
Maximum number of terminology files per `TranslateText` or `StartTextTranslationJob` request.	1
Transactions per second for the ImportTerminology API action	5
Transactions per second for the GetTerminology API action	10
Transactions per second for the ListTerminologies API action	10
Transactions per second for the DeleteTerminology API action	5

Parallel data quotas

Description	Limit
Maximum number of parallel data resources per AWS account per AWS Region	1000
Maximum parallel data input file size	5 GB
Maximum number of source languages in a parallel data resource	1
Maximum size of a single segment or record in a parallel data input file	1000 bytes
Maximum number of concurrent create or update operations for parallel data resources	1
Transactions per second for the CreateParallelData API action	5
Transactions per second for the GetParallelData API action	10
Transactions per second for the ListParallelData API action	10
Transactions per second for the UpdateParallelData API action	5
Transactions per second for the DeleteParallelData API action	5

Document history for Amazon Translate

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

Change	Description	Date
New feature: Brevity	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 Using brevity in Amazon Translate.	October 31, 2023
Language auto-detection for document input to real-time translations	You can now use language auto-detection when you input a document to real-time translations (console or API). For more information, see Real-time translations.	August 3, 2023
Word (.docx) files as input to real-time translations	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 Real-time translations.	July 17, 2023
Enhancements to custom terminology	Translate now supports enhancements to the custom terminology feature that improve translation fluency and accuracy. For more information, see Customizing	June 30, 2023

	your translations with custom terminology.	
Text or HTML files as input to real-time translations	You can now use text files or HTML files as input to real-time translations (console or API). For more information, see Real-time translations.	May 23, 2023
New action allowed in the TranslateReadOnly policy	Amazon Translate now allows the `TranslateDocument` action in the TranslateReadOnly managed policy. For more information, see AWS managed policy: TranslateReadOnly.	May 23, 2023
Translate now supports additional regions for asynchronous batch processing.	Translate now supports additional regions for asynchronous batch processing. For more information, see Asynchronous batch processing with Amazon Translate.	March 28, 2023
Increased input size for real-time translations	You can now input up to 10,000 characters for real-time translations. For more information, see Getting started in Amazon Translate (console).	December 16, 2022
Support for nested input folders for batch mode	You can now provide nested input folders to batch translation jobs. For more information, see Running a batch translation job in Amazon Translate.	November 18, 2022

Support for auto-language detection for batch mode	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 Running a batch translation job in Amazon Translate.	November 18, 2022
Support for multiple target languages	You can now specify multiple target languages in batch translation jobs. For more information, see Running a batch translation job in Amazon Translate.	October 10, 2022
Support for tags	You can now tag **ParallelData** and **Custom Terminology** resources in Amazon Translate. For more information, see Tagging your resources in Amazon Translate.	October 6, 2022
Formality support for additional languages	You can now set the translation formality level for Dutch, Korean, and Mexican Spanish in Amazon Translate. For more information, see Setting formality in Amazon Translate.	October 5, 2022

Separate API Reference	The Amazon Translate API Reference is now a separate document from the Developer Guide. For more informati on, see Amazon Translate API Reference.	August 25, 2022
New feature	You can now set the formality level for your translation output. For more informati on, see Setting formality in Amazon Translate.	February 22, 2022
New feature	You can now mask profane words and phrases in your translation output. For more information, see Masking profane words and phrases in Amazon Translate.	November 24, 2021
AWS PrivateLink support	You can now establish a private connection between your VPC and Amazon Translate by using AWS PrivateLink. For more information, see Amazon Translate and interface VPC endpoints (AWS PrivateLink).	November 24, 2021

Parallel data update	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 informati on about parallel data, see Customizing your translati ons with parallel data (Active Custom Translation).	November 15, 2021
Custom terminology direction ality	You can now create multi-directional terminology, in which any language can be the source language or a target language. For more information, see Creating a custom terminology.	November 11, 2021
New languages	Amazon Translate now supports the following languages: Irish, Marathi, Portuguese (Portugal), and Punjabi. For all of the languages that Amazon Translate supports, see Supported languages and language codes.	November 10, 2021

New custom encryption settings	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 Running a batch translation job	November 5, 2021
New file format support	Amazon Translate now supports XML Localization Interchange File Format (XLIFF) files for asynchron ous batch processing. For all supported formats, see Supported file formats.	June 9, 2021
EventBridge integration	Amazon Translate now sends events to Amazon EventBrid ge to notify you about changes that affect your translation jobs and parallel data resources. For more information, see Monitoring Amazon Translate events with Amazon EventBridge.	June 4, 2021
New quota	Amazon Translate now supports up to 1000 queued batch translation jobs. For all Amazon Translate quotas, see Guidelines and limits.	April 23, 2021

Quota increase	The maximum size for a parallel data input file has increased from 1 MB to 5 MB. For all Amazon Translate quotas, see Guidelines and limits.	March 31, 2021
New languages	Amazon Translate now supports the following languages: Armenian, Catalan, Farsi (Persian), Filipino Tagalog, Gujarati, Haitian Creole, Icelandic, Kannada, Kazakh, Lithuania n, Macedonian, Malayalam, Maltese, Mongolian, Sinhala, Telugu, Uzbek, and Welsh. For all of the languages that Amazon Translate supports, see Supported languages and language codes.	November 23, 2020

New feature	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 Customizing your translations with parallel data (Active Custom Translation).	November 23, 2020
New file format support	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.	July 29, 2020
New language	Amazon Translate now supports the Spanish (Mexico) language for translation. For all supported languages, see Supported languages and language codes.	April 30, 2020

New region	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.	April 20, 2020
New feature	Amazon Translate adds asynchronous batch translation functionality. For more information, see Asynchronous batch processing.	December 23, 2019
New regions	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*.	November 25, 2019

New languages	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.	November 25, 2019
New languages	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.	October 3, 2019
New feature	Amazon Translate adds FedRAMP compliance. For more information, see Compliance.	July 31, 2019
New feature	Amazon Translate adds SOC compliance. For more information, see Compliance.	May 30, 2019

New regions	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 AWS Region Table or AWS Regions and Endpoints in the *Amazon Web Services General Reference*.	May 8, 2019
New languages	Amazon Translate adds new languages for translation: Hindi, Malay, Norwegian , and Persian. For a list of the language combinati ons that Amazon Translate can translate directly, see Supported languages.	May 6, 2019
New region	Amazon Translate adds support for the EU (Frankfur t) and Asia Pacific (Seoul) 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*.	February 28, 2019
New feature	Amazon Translate adds PCI compliance. For more information, see Compliance.	December 12, 2018

New feature	Amazon Translate adds four new APIs and the custom terminology feature to give you more control over your translation. By using a custom terminology with your translation requests, you can make sure that your brand names, character names, model names, and other unique content is translate d exactly the way you want it, every time, regardless of the standard translation or context. For more informati on, see Custom terminology.	November 27, 2018
New languages	Amazon Translate now translates documents in the following languages : Danish, Dutch, Finnish, Hebrew, Indonesian, Korean, Polish, and Swedish. Amazon Translate continues to improve direct translation by significantly reducing the number of unsupport ed language pairs. For the language combinations that Amazon Translate can translate directly, see Supported languages.	November 20, 2018

New feature	Amazon Translate adds direct translation between supported languages other than English. For the language combinations that Amazon Translate can translate directly, see Supported languages.	October 29, 2018
New feature	Amazon Translate adds HIPAA compliance. For more information, see Compliance.	October 25, 2018
New feature	Amazon Translate adds multiple new languages for translation: Chinese (Traditional), Czech, Italian, Japanese, Russian, and Turkish. For a list of languages that Amazon Translate supports, see Supported languages.	July 17, 2018
New feature	Amazon Translate adds support for automatic source language detection. For more information, see How Amazon Translate works.	April 4, 2018
New guide	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*.