Content Moderator documentation

Azure Content Moderator is deprecated as of February 2024 and will be retired by February 2027. It is replaced by Azure AI Content Safety, which offers advanced AI features and enhanced performance. Azure AI Content Safety is a comprehensive solution designed to detect harmful user-generated and AI-generated content in applications and services. Azure AI Content Safety is suitable for many scenarios such as online marketplaces, gaming companies, social messaging platforms, enterprise media companies, and K-12 education solution providers.

Azure AI Content Safety

GET STARTED

Go to Azure AI Content Safety

About Content Moderator

OVERVIEW

What is Content Moderator?

TRAINING

Microsoft Learn training

VIDEO

Video moderation with Content Moderator

Analyze image content

CONCEPT

Image moderation

QUICKSTART

Using the .NET SDK

Using the Python SDK

Using the Java SDK

Using the REST API

Use custom image lists

QUICKSTART

Using the .NET SDK

Using the Python SDK

HOW-TO GUIDE

Using the .NET SDK

Analyze text content

CONCEPT

Text moderation

QUICKSTART

Using the .NET SDK

Using the Python SDK

Using the REST API

Use custom terms lists

QUICKSTART

Using the .NET SDK

Using the Python SDK

HOW-TO GUIDE

Using the .NET SDK

Analyze video content

HOW-TO GUIDE

Using the .NET SDK

VIDEO

Video moderation with Content Moderator

Reference

REFERENCE

Content Moderator API

.NET SDK

Python SDK

Java SDK

Node.js SDK

Go SDK

Azure PowerShell

Azure Command-Line Interface (CLI)

Help and feedback

REFERENCE

Support and help options

What is Azure Content Moderator?

Article • 01/18/2024

ⓘ Important

Azure Content Moderator is being deprecated in February 2024, and will be retired by February 2027. It is being replaced by **Azure AI Content Safety**, which offers advanced AI features and enhanced performance.

Azure AI Content Safety is a comprehensive solution designed to detect harmful user-generated and AI-generated content in applications and services. Azure AI Content Safety is suitable for many scenarios such as online marketplaces, gaming companies, social messaging platforms, enterprise media companies, and K-12 education solution providers. Here's an overview of its features and capabilities:

- **Text and Image Detection APIs**: Scan text and images for sexual content, violence, hate, and self-harm with multiple severity levels.
- **Content Safety Studio**: An online tool designed to handle potentially offensive, risky, or undesirable content using our latest content moderation ML models. It provides templates and customized workflows that enable users to build their own content moderation systems.
- **Language support**: Azure AI Content Safety supports more than 100 languages and is specifically trained on English, German, Japanese, Spanish, French, Italian, Portuguese, and Chinese.

Azure AI Content Safety provides a robust and flexible solution for your content moderation needs. By switching from Content Moderator to Azure AI Content Safety, you can take advantage of the latest tools and technologies to ensure that your content is always moderated to your exact specifications.

Learn more about Azure AI Content Safety and explore how it can elevate your content moderation strategy.

Azure Content Moderator is an AI service that lets you handle content that is potentially offensive, risky, or otherwise undesirable. It includes the AI-powered content moderation service which scans text, image, and videos and applies content flags automatically.

You may want to build content filtering software into your app to comply with regulations or maintain the intended environment for your users.

This documentation contains the following article types:

- **Quickstarts** are getting-started instructions to guide you through making requests to the service.
- **How-to guides** contain instructions for using the service in more specific or customized ways.
- **Concepts** provide in-depth explanations of the service functionality and features.

For a more structured approach, follow a Training module for Content Moderator.

- Introduction to Content Moderator
- Classify and moderate text with Azure Content Moderator

Where it's used

The following are a few scenarios in which a software developer or team would require a content moderation service:

- Online marketplaces that moderate product catalogs and other user-generated content.
- Gaming companies that moderate user-generated game artifacts and chat rooms.
- Social messaging platforms that moderate images, text, and videos added by their users.
- Enterprise media companies that implement centralized moderation for their content.
- K-12 education solution providers filtering out content that is inappropriate for students and educators.

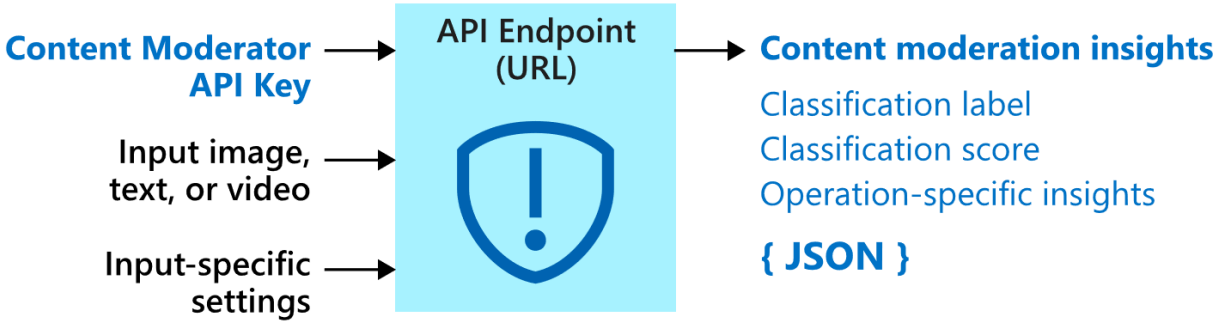
> **Important**
>
> You cannot use Content Moderator to detect illegal child exploitation images. However, qualified organizations can use the **PhotoDNA Cloud Service** to screen for this type of content.

What it includes

The Content Moderator service consists of several web service APIs available through both REST calls and a .NET SDK.

Moderation APIs

The Content Moderator service includes Moderation APIs, which check content for material that is potentially inappropriate or objectionable.



The following table describes the different types of moderation APIs.

Expand table

API group	Description
Text moderation	Scans text for offensive content, sexually explicit or suggestive content, profanity, and personal data.
Custom term lists	Scans text against a custom list of terms along with the built-in terms. Use custom lists to block or allow content according to your own content policies.
Image moderation	Scans images for adult or racy content, detects text in images with the Optical Character Recognition (OCR) capability, and detects faces.
Custom image lists	Scans images against a custom list of images. Use custom image lists to filter out instances of commonly recurring content that you don't want to classify again.
Video moderation	Scans videos for adult or racy content and returns time markers for said content.

Data privacy and security

As with all of the Azure AI services, developers using the Content Moderator service should be aware of Microsoft's policies on customer data. See the Azure AI services page on the Microsoft Trust Center to learn more.

Next steps

- Complete a client library or REST API quickstart to implement the basic scenarios in code.

Language support for Content Moderator API

Article • 01/18/2024

> **Note**
>
> For the **language** parameter, assign `eng` or leave it empty to see the machine-assisted **classification** response (preview feature). **This feature supports English only**.
>
> For **profanity terms** detection, use the **ISO 639-3 code** of the supported languages listed in this article, or leave it empty.

Expand table

Language	Language detection	Profanity	OCR	Auto-correction
Afrikaans		✓		
Albanian		✓		
Amharic		✓		
Arabic		✓	✓	✓
Arabic (Romanized)	✓			
Armenian		✓		
Assamese		✓		
Azerbaijani		✓		
Bangla - Bangladesh		✓		
Balinese	✓			
Basque		✓		
Belarusian		✓		
Bengali	✓			
Bengali - India		✓		

Language	Language detection	Profanity	OCR	Auto-correction
Bosnian - Cyrillic		✓		
Bosnian - Latin		✓		
Buginese	✓			
Buhid	✓			
Bulgarian		✓		
Breton (non-GeoPol)		✓		
Carian	✓			
Catalan		✓		
Central Kurdish		✓		
Cherokee		✓		
Chinese (Simplified)	✓	✓	✓	
Chinese (Traditional)	✓		✓	
Chinese (Traditional) - Hong Kong SAR		✓		
Chinese (Traditional) - Taiwan		✓		
Church (Slavic)	✓			
Coptic	✓			
Croatian		✓		
Czech	✓	✓	✓	
Danish		✓	✓	✓
Dari	✓	✓		
Dhivehi	✓			
Dutch	✓	✓	✓	✓
English		✓	✓	✓
English (Creole)	✓			
Estonian		✓		

Language	Language detection	Profanity	OCR	Auto-correction
Filipino		✓		
Finnish		✓	✓	✓
French	✓	✓	✓	✓
Georgian		✓		
German	✓	✓	✓	
Greek	✓	✓		
Greek (modern)			✓	✓
Gujarati		✓		
Haitian	✓			
Hausa		✓		
Hebrew	✓	✓		
Hindi	✓	✓		
Hmong	✓			
Hungarian	✓	✓	✓	
Icelandic		✓		
Igbo		✓		
Indonesian		✓		
Inuktitut		✓		
Irish		✓		
isiXhosa		✓		
isiZulu		✓		
Italian	✓	✓	✓	✓
Japanese	✓	✓	✓	
Kannada		✓		
Kazakh		✓		
Khmer		✓		

Language	Language detection	Profanity	OCR	Auto-correction
K'iche		✓		
Kinyarwanda		✓		
Kiswahili		✓		
Konkani		✓		
Korean	✓	✓	✓	✓
Kurdish (Arabic)	✓			
Kurdish (Latin)	✓			
Kyrgyz		✓		
Lao		✓		
Latvian		✓		
Lepcha	✓			
Limbu	✓			
Lithuanian		✓		
Lu	✓			
Luxembourgish		✓		
Lycian	✓			
Lydian	✓			
Macedonian		✓		
Malay		✓		
Malayalam		✓		
Maltese		✓		
Maori		✓		
Marathi		✓		
Mongolian		✓		
Mycenaean (Greek)	✓			
Nepali		✓		

Language	Language detection	Profanity	OCR	Auto-correction
Nko	✓			
Norwegian			✓	✓
Norwegian (Bokmal)	✓	✓		
Norwegian (Nynorsk)	✓	✓		
Odia		✓		
Pashto	✓	✓		
Persian	✓	✓		
Polish	✓	✓	✓	✓
Portuguese - Brazil	✓	✓	✓	✓
Portuguese - Portugal	✓	✓	✓	✓
Pulaar		✓		
Punjabi	✓	✓		
Punjabi (Pakistan)		✓		
Quechua (Peru)		✓		
Rejang	✓			
Romanian		✓	✓	✓
Russian	✓	✓	✓	✓
Santali	✓			
Sasak	✓			
Saurashtra	✓			
Serbian (Cyrillic)	✓	✓	✓	
Serbian (Cyrillic, Bosnia and Herzegovina)		✓		
Serbian (Latin)	✓	✓	✓	
Sesotho		✓		
Sesotho sa Leboa		✓		

Language	Language detection	Profanity	OCR	Auto-correction
Setswana		✓		
Sindhi		✓		
Sinhala	✓	✓		
Slovak		✓	✓	✓
Slovenian	✓	✓		
Spanish	✓	✓	✓	✓
Swedish	✓	✓	✓	
Sylheti	✓			
Syriac	✓			
Tagbanwa	✓			
Tai (Nua)	✓			
Tajik		✓		
Tamashek	✓			
Tamil		✓		
Tatar		✓		
Telugu		✓		
Thai		✓		
Tigrinya		✓		
Turkish	✓	✓	✓	✓
Turkmen		✓		
Ugaritic	✓			
Ukrainian		✓		
Urdu		✓		
Uyghur		✓		
Uzbek (Cyrillic)	✓	✓		
Uzbek (Latin)	✓	✓		

Language	Language detection	Profanity	OCR	Auto-correction
Valencian		✔		
Vai	✔			
Vietnamese		✔		
Wolof		✔		
Yi	✔			
Yoruba		✔		
Zhuang, Chuang	✔			

Quickstart: Use the Content Moderator client library

Article • 01/18/2024

ⓘ Important

Azure Content Moderator is being deprecated in February 2024, and will be retired by February 2027. It is being replaced by **Azure AI Content Safety**, which offers advanced AI features and enhanced performance.

Azure AI Content Safety is a comprehensive solution designed to detect harmful user-generated and AI-generated content in applications and services. Azure AI Content Safety is suitable for many scenarios such as online marketplaces, gaming companies, social messaging platforms, enterprise media companies, and K-12 education solution providers. Here's an overview of its features and capabilities:

- **Text and Image Detection APIs**: Scan text and images for sexual content, violence, hate, and self-harm with multiple severity levels.
- **Content Safety Studio**: An online tool designed to handle potentially offensive, risky, or undesirable content using our latest content moderation ML models. It provides templates and customized workflows that enable users to build their own content moderation systems.
- **Language support**: Azure AI Content Safety supports more than 100 languages and is specifically trained on English, German, Japanese, Spanish, French, Italian, Portuguese, and Chinese.

Azure AI Content Safety provides a robust and flexible solution for your content moderation needs. By switching from Content Moderator to Azure AI Content Safety, you can take advantage of the latest tools and technologies to ensure that your content is always moderated to your exact specifications.

Learn more about Azure AI Content Safety and explore how it can elevate your content moderation strategy.

Get started with the Azure Content Moderator client library for .NET. Follow these steps to install the NuGet package and try out the example code for basic tasks.

Content Moderator is an AI service that lets you handle content that is potentially offensive, risky, or otherwise undesirable. Use the AI-powered content moderation

service to scan text, image, and videos and apply content flags automatically. Build content filtering software into your app to comply with regulations or maintain the intended environment for your users.

Use the Content Moderator client library for .NET to:

- Moderate text
- Moderate images

Reference documentation | Library source code | Package (NuGet) | Samples

Prerequisites

- Azure subscription - Create one for free
- The Visual Studio IDE or current version of .NET Core.
- Once you have your Azure subscription, create a Content Moderator resource in the Azure portal to get your key and endpoint. Wait for it to deploy and click the **Go to resource** button.
  - You will need the key and endpoint from the resource you create to connect your application to Content Moderator. You'll paste your key and endpoint into the code below later in the quickstart.
  - You can use the free pricing tier (`F0`) to try the service, and upgrade later to a paid tier for production.

Setting up

Create a new C# application

Visual Studio IDE

Using Visual Studio, create a new .NET Core application.

Install the client library

Once you've created a new project, install the client library by right-clicking on the project solution in the **Solution Explorer** and selecting **Manage NuGet Packages**. In the package manager that opens select **Browse**, check **Include prerelease**, and search for `Microsoft.Azure.CognitiveServices.ContentModerator`. Select version `2.0.0`, and then **Install**.

> **Tip**
>
> Want to view the whole quickstart code file at once? You can find it on **GitHub**, which contains the code examples in this quickstart.

From the project directory, open the *Program.cs* file in your preferred editor or IDE. Add the following `using` statements:

C#

```
using Microsoft.Azure.CognitiveServices.ContentModerator;
using Microsoft.Azure.CognitiveServices.ContentModerator.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Threading;
```

In the **Program** class, create variables for your resource's key and endpoint.

> **Important**
>
> Go to the Azure portal. If the Content Moderator resource you created in the **Prerequisites** section deployed successfully, click the **Go to Resource** button under **Next Steps**. You can find your key and endpoint in the resource's **key and endpoint** page, under **resource management**.

C#

```
// Your Content Moderator subscription key is found in your Azure portal
resource on the 'Keys' page.
private static readonly string SubscriptionKey =
"PASTE_YOUR_CONTENT_MODERATOR_SUBSCRIPTION_KEY_HERE";
// Base endpoint URL. Found on 'Overview' page in Azure resource. For
example: https://westus.api.cognitive.microsoft.com
private static readonly string Endpoint =
"PASTE_YOUR_CONTENT_MODERATOR_ENDPOINT_HERE";
```

> **Important**
>
> Remember to remove the key from your code when you're done, and never post it publicly. For production, use a secure way of storing and accessing your credentials

like **Azure Key Vault**. See the Azure AI services **security** article for more information.

In the application's `main()` method, add calls for the methods used in this quickstart. You will create these later.

C#

```
// Create an image review client
ContentModeratorClient clientImage = Authenticate(SubscriptionKey,
Endpoint);
// Create a text review client
ContentModeratorClient clientText = Authenticate(SubscriptionKey, Endpoint);
// Create a human reviews client
ContentModeratorClient clientReviews = Authenticate(SubscriptionKey,
Endpoint);
```

C#

```
// Moderate text from text in a file
ModerateText(clientText, TextFile, TextOutputFile);
```

C#

```
// Moderate images from list of image URLs
ModerateImages(clientImage, ImageUrlFile, ImageOutputFile);
```

Object model

The following classes handle some of the major features of the Content Moderator .NET client library.

Expand table

Name	Description
ContentModeratorClient	This class is needed for all Content Moderator functionality. You instantiate it with your subscription information, and you use it to produce instances of other classes.
ImageModeration	This class provides the functionality for analyzing images for adult content, personal information, or human faces.
TextModeration	This class provides the functionality for analyzing text for language, profanity, errors, and personal information.

Code examples

These code snippets show you how to do the following tasks with the Content Moderator client library for .NET:

- Authenticate the client
- Moderate text
- Moderate images

Authenticate the client

In a new method, instantiate client objects with your endpoint and key.

C#

```
public static ContentModeratorClient Authenticate(string key, string
endpoint)
{
    ContentModeratorClient client = new ContentModeratorClient(new
ApiKeyServiceClientCredentials(key));
    client.Endpoint = endpoint;

    return client;
}
```

Moderate text

The following code uses a Content Moderator client to analyze a body of text and print the results to the console. In the root of your **Program** class, define input and output files:

C#

```
// TEXT MODERATION
// Name of the file that contains text
private static readonly string TextFile = "TextFile.txt";
// The name of the file to contain the output from the evaluation.
private static string TextOutputFile = "TextModerationOutput.txt";
```

Then at the root of your project, add a *TextFile.txt* file. Add your own text to this file, or use the following sample text:

```
```

```
Is this a grabage email abcdef@abcd.com, phone: 4255550111, IP:
255.255.255.255, 1234 Main Boulevard, Panapolis WA 96555.
<offensive word> is the profanity here. Is this information PII? phone
4255550111
```

Then define the text moderation method somewhere in your **Program** class:

C#

```
/*
 * TEXT MODERATION
 * This example moderates text from file.
 */
public static void ModerateText(ContentModeratorClient client, string
inputFile, string outputFile)
{
    Console.WriteLine("--------------------------------------------------------
---------");
    Console.WriteLine();
    Console.WriteLine("TEXT MODERATION");
    Console.WriteLine();
    // Load the input text.
    string text = File.ReadAllText(inputFile);

    // Remove carriage returns
    text = text.Replace(Environment.NewLine, " ");
    // Convert string to a byte[], then into a stream (for parameter in
ScreenText()).
    byte[] textBytes = Encoding.UTF8.GetBytes(text);
    MemoryStream stream = new MemoryStream(textBytes);

    Console.WriteLine("Screening {0}...", inputFile);
    // Format text

    // Save the moderation results to a file.
    using (StreamWriter outputWriter = new StreamWriter(outputFile, false))
    {
        using (client)
        {
            // Screen the input text: check for profanity, classify the text
into three categories,
            // do autocorrect text, and check for personally identifying
information (PII)
            outputWriter.WriteLine("Autocorrect typos, check for matching
terms, PII, and classify.");

            // Moderate the text
            var screenResult =
client.TextModeration.ScreenText("text/plain", stream, "eng", true, true,
null, true);
            outputWriter.WriteLine(JsonConvert.SerializeObject(screenResult,
Formatting.Indented));
```

```
        }

        outputWriter.Flush();
        outputWriter.Close();
    }

    Console.WriteLine("Results written to {0}", outputFile);
    Console.WriteLine();
}
```

Moderate images

The following code uses a Content Moderator client, along with an ImageModeration object, to analyze remote images for adult and racy content.

> **Note**
>
> You can also analyze the content of a local image. See the **reference documentation** for methods and operations that work with local images.

Get sample images

Define your input and output files at the root of your **Program** class:

C#

```
// IMAGE MODERATION
//The name of the file that contains the image URLs to evaluate.
private static readonly string ImageUrlFile = "ImageFiles.txt";
// The name of the file to contain the output from the evaluation.
private static string ImageOutputFile = "ImageModerationOutput.json";
```

Then create the input file, *ImageFiles.txt*, at the root of your project. In this file, you add the URLs of images to analyze—one URL on each line. You can use the following sample images:

```
https://moderatorsampleimages.blob.core.windows.net/samples/sample2.jpg
https://moderatorsampleimages.blob.core.windows.net/samples/sample5.png
```

Define helper class

Add the following class definition within the **Program** class. This inner class will handle image moderation results.

C#

```
// Contains the image moderation results for an image,
// including text and face detection results.
public class EvaluationData
{
    // The URL of the evaluated image.
    public string ImageUrl;

    // The image moderation results.
    public Evaluate ImageModeration;

    // The text detection results.
    public OCR TextDetection;

    // The face detection results;
    public FoundFaces FaceDetection;
}
```

Define the image moderation method

The following method iterates through the image URLs in a text file, creates an **EvaluationData** instance, and analyzes the image for adult/racy content, text, and human faces. Then it adds the final **EvaluationData** instance to a list and writes the complete list of returned data to the console.

Iterate through images

C#

```
/*
 * IMAGE MODERATION
 * This example moderates images from URLs.
 */
public static void ModerateImages(ContentModeratorClient client, string
urlFile, string outputFile)
{
    Console.WriteLine("------------------------------------------------------
---------");
    Console.WriteLine();
    Console.WriteLine("IMAGE MODERATION");
    Console.WriteLine();
    // Create an object to store the image moderation results.
    List<EvaluationData> evaluationData = new List<EvaluationData>();
```

```
    using (client)
    {
        // Read image URLs from the input file and evaluate each one.
        using (StreamReader inputReader = new StreamReader(urlFile))
        {
            while (!inputReader.EndOfStream)
            {
                string line = inputReader.ReadLine().Trim();
                if (line != String.Empty)
                {
                    Console.WriteLine("Evaluating {0}...",
Path.GetFileName(line));
                    var imageUrl = new BodyModel("URL", line.Trim());
```

Analyze content

For more information on the image attributes that Content Moderator screens for, see the Image moderation concepts guide.

C#

```
            var imageData = new EvaluationData
            {
                ImageUrl = imageUrl.Value,

                // Evaluate for adult and racy content.
                ImageModeration =
                client.ImageModeration.EvaluateUrlInput("application/json",
imageUrl, true)
            };
            Thread.Sleep(1000);

            // Detect and extract text.
            imageData.TextDetection =
                client.ImageModeration.OCRUrlInput("eng",
"application/json", imageUrl, true);
            Thread.Sleep(1000);

            // Detect faces.
            imageData.FaceDetection =
                client.ImageModeration.FindFacesUrlInput("application/json",
imageUrl, true);
            Thread.Sleep(1000);

            // Add results to Evaluation object
            evaluationData.Add(imageData);
        }
    }
}
```

Write moderation results to file

C#

```
        // Save the moderation results to a file.
        using (StreamWriter outputWriter = new StreamWriter(outputFile,
false))
        {
            outputWriter.WriteLine(JsonConvert.SerializeObject(
                evaluationData, Formatting.Indented));

            outputWriter.Flush();
            outputWriter.Close();
        }
        Console.WriteLine();
        Console.WriteLine("Image moderation results written to output file:
" + outputFile);
        Console.WriteLine();
    }
}
```

Run the application

Visual Studio IDE

Run the application by clicking the **Debug** button at the top of the IDE window.

Clean up resources

If you want to clean up and remove an Azure AI services subscription, you can delete the resource or resource group. Deleting the resource group also deletes any other resources associated with it.

- Portal
- Azure CLI

Next steps

In this quickstart, you learned how to use the Content Moderator .NET library to do moderation tasks. Next, learn more about the moderation of images or other media by reading a conceptual guide.

Image moderation concepts

Content Moderator .NET SDK samples

Article • 01/18/2024

The following list includes links to the code samples built using the Azure Content Moderator SDK for .NET.

- **Image moderation**: Evaluate an image for adult and racy content, text, and faces. See the .NET SDK quickstart.
- **Custom images**: Moderate with custom image lists. See the .NET SDK quickstart.

> **Note**
>
> There is a maximum limit of **5 image lists** with each list to **not exceed 10,000 images**.

- **Text moderation**: Screen text for profanity and personal data. See the .NET SDK quickstart.
- **Custom terms**: Moderate with custom term lists. See the .NET SDK quickstart.

> **Note**
>
> There is a maximum limit of **5 term lists** with each list to **not exceed 10,000 terms**.

- **Video moderation**: Scan a video for adult and racy content and get results. See quickstart.

See all .NET samples at the Content Moderator .NET samples on GitHub.

Content Moderator REST samples in C#

Article • 01/18/2024

The following list includes links to code samples built using the Azure Content Moderator API.

- Image moderation
- Text moderation
- Video moderation

For walkthroughs of these samples, check out the on-demand webinar .

Analyze video content for objectionable material in C#

Article • 01/18/2024

This article provides information and code samples to help you get started using the Content Moderator SDK for .NET to scan video content for adult or racy content.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

- Any edition of Visual Studio 2015 or 2017

Set up Azure resources

The Content Moderator's video moderation capability is available as a free public preview **media processor** in Azure Media Services (AMS). Azure Media Services is a specialized Azure service for storing and streaming video content.

Create an Azure Media Services account

Follow the instructions in Create an Azure Media Services account to subscribe to AMS and create an associated Azure storage account. In that storage account, create a new Blob storage container.

Create a Microsoft Entra application

Navigate to your new AMS subscription in the Azure portal and select **API access** from the side menu. Select **Connect to Azure Media Services with service principal**. Note the value in the **REST API endpoint** field; you will need this later.

In the **Microsoft Entra app** section, select **Create New** and name your new Microsoft Entra application registration (for example, "VideoModADApp"). Select **Save** and wait a few minutes while the application is configured. Then, you should see your new app registration under the **Microsoft Entra app** section of the page.

Select your app registration and click the **Manage application** button below it. Note the value in the **Application ID** field; you will need this later. Select **Settings** > **Keys**, and

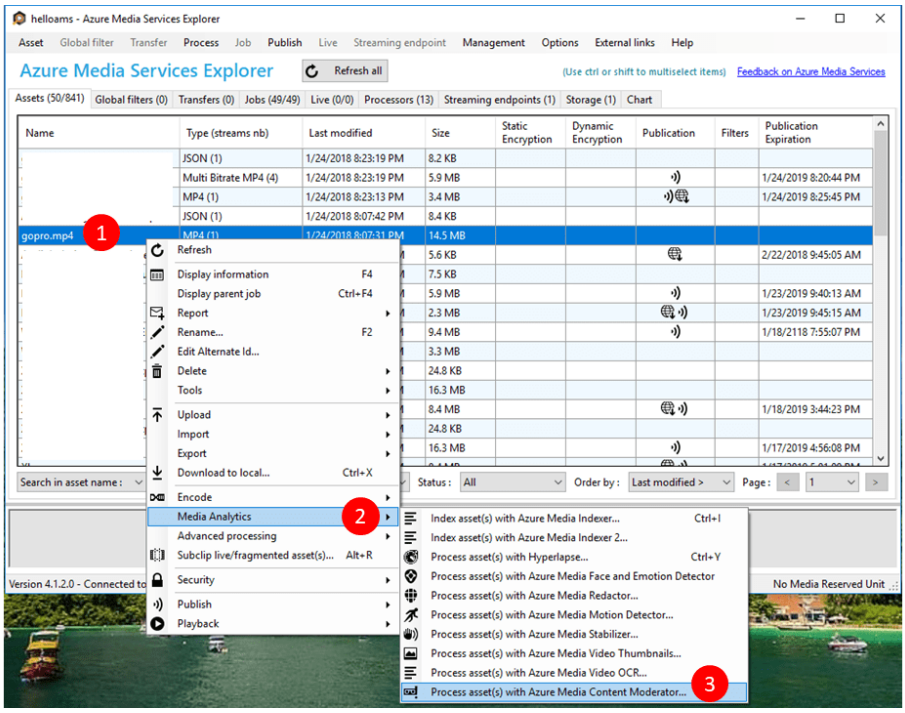
enter a description for a new key (such as "VideoModKey"). Select **Save**, and then notice the new key value. Copy this string and save it somewhere secure.

For a more thorough walkthrough of the above process, See Get started with Microsoft Entra authentication.

Once you've done this, you can use the video moderation media processor in two different ways.

Use Azure Media Services Explorer

The Azure Media Services Explorer is a user-friendly frontend for AMS. Use it to browse your AMS account, upload videos, and scan content with the Content Moderator media processor. Download and install it from GitHub, or see the Azure Media Services Explorer blog post for more information.



Create the Visual Studio project

1. In Visual Studio, create a new **Console app (.NET Framework)** project and name it **VideoModeration**.
2. If there are other projects in your solution, select this one as the single startup project.
3. Get the required NuGet packages. Right-click on your project in the Solution Explorer and select **Manage NuGet Packages**; then find and install the following packages:

- windowsazure.mediaservices
- windowsazure.mediaservices.extensions

Add video moderation code

Next, you'll copy and paste the code from this guide into your project to implement a basic content moderation scenario.

Update the program's using statements

Add the following `using` statements to the top of your *Program.cs* file.

C#

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.WindowsAzure.MediaServices.Client;
using System.IO;
using System.Threading;
using Microsoft.WindowsAzure.Storage.Blob;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using System.Collections.Generic;
```

Set up resource references

Add the following static fields to the **Program** class in *Program.cs*. These fields hold the information necessary for connecting to your AMS subscription. Fill them in with the values you got in the steps above. Note that `CLIENT_ID` is the **Application ID** value of your Microsoft Entra app, and `CLIENT_SECRET` is the value of the "VideoModKey" that you created for that app.

C#

```
// declare constants and globals
private static CloudMediaContext _context = null;
private static CloudStorageAccount _StorageAccount = null;

// Azure Media Services (AMS) associated Storage Account, Key, and the
Container that has
// a list of Blobs to be processed.
static string STORAGE_NAME = "YOUR AMS ASSOCIATED BLOB STORAGE NAME";
static string STORAGE_KEY = "YOUR AMS ASSOCIATED BLOB STORAGE KEY";
static string STORAGE_CONTAINER_NAME = "YOUR BLOB CONTAINER FOR VIDEO
```

```
FILES";

private static StorageCredentials _StorageCredentials = null;

// Azure Media Services authentication.
private const string AZURE_AD_TENANT_NAME = "microsoft.onmicrosoft.com";
private const string CLIENT_ID = "YOUR CLIENT ID";
private const string CLIENT_SECRET = "YOUR CLIENT SECRET";

// REST API endpoint, for example
"https://accountname.restv2.westcentralus.media.azure.net/API".
private const string REST_API_ENDPOINT = "YOUR API ENDPOINT";

// Content Moderator Media Processor Nam
private const string MEDIA_PROCESSOR = "Azure Media Content Moderator";

// Input and Output files in the current directory of the executable
private const string INPUT_FILE = "VIDEO FILE NAME";
private const string OUTPUT_FOLDER = "";

// JSON settings file
private static readonly string CONTENT_MODERATOR_PRESET_FILE =
"preset.json";
```

> **Important**
>
> Remember to remove the keys from your code when you're done, and never post them publicly. For production, use a secure way of storing and accessing your credentials like **Azure Key Vault**. See the Azure AI services **security** article for more information.

If you wish to use a local video file (simplest case), add it to the project and enter its path as the `INPUT_FILE` value (relative paths are relative to the execution directory).

You will also need to create the *preset.json* file in the current directory and use it to specify a version number. For example:

JSON

```
{
    "version": "2.0"
}
```

Load the input video(s)

The **Main** method of the **Program** class will create an Azure Media Context and then an Azure Storage Context (in case your videos are in blob storage). The remaining code scans a video from a local folder, blob, or multiple blobs within an Azure storage container. You can try all options by commenting out the other lines of code.

C#

```
// Create Azure Media Context
CreateMediaContext();

// Create Storage Context
CreateStorageContext();

// Use a file as the input.
IAsset asset = CreateAssetfromFile();

// -- OR ---

// Or a blob as the input
// IAsset asset =
CreateAssetfromBlob((CloudBlockBlob)GetBlobsList().First());

// Then submit the asset to Content Moderator
RunContentModeratorJob(asset);

//-- OR ----

// Just run the content moderator on all blobs in a list (from a Blob
Container)
// RunContentModeratorJobOnBlobs();
```

Create an Azure Media Context

Add the following method to the **Program** class. This uses your AMS credentials to allow communication with AMS.

C#

```
// Creates a media context from azure credentials
static void CreateMediaContext()
{
    // Get Azure AD credentials
    var tokenCredentials = new AzureAdTokenCredentials(AZURE_AD_TENANT_NAME,
        new AzureAdClientSymmetricKey(CLIENT_ID, CLIENT_SECRET),
        AzureEnvironments.AzureCloudEnvironment);

    // Initialize an Azure AD token
    var tokenProvider = new AzureAdTokenProvider(tokenCredentials);

    // Create a media context
```

```
    _context = new CloudMediaContext(new Uri(REST_API_ENDPOINT),
tokenProvider);
}
```

Add the code to create an Azure Storage Context

Add the following method to the **Program** class. You use the Storage Context, created from your storage credentials, to access your blob storage.

C#

```
// Creates a storage context from the AMS associated storage name and key
static void CreateStorageContext()
{
    // Get a reference to the storage account associated with a Media
Services account.
    if (_StorageCredentials == null)
    {
        _StorageCredentials = new StorageCredentials(STORAGE_NAME,
STORAGE_KEY);
    }
    _StorageAccount = new CloudStorageAccount(_StorageCredentials, false);
}
```

Add the code to create Azure Media Assets from local file and blob

The Content Moderator media processor runs jobs on **Assets** within the Azure Media Services platform. These methods create the Assets from a local file or an associated blob.

C#

```
// Creates an Azure Media Services Asset from the video file
static IAsset CreateAssetfromFile()
{
    return _context.Assets.CreateFromFile(INPUT_FILE,
AssetCreationOptions.None); ;
}

// Creates an Azure Media Services asset from your blog storage
static IAsset CreateAssetfromBlob(CloudBlockBlob Blob)
{
    // Create asset from the FIRST blob in the list and return it
    return _context.Assets.CreateFromBlob(Blob, _StorageCredentials,
```

```
AssetCreationOptions.None);
}
```

Add the code to scan a collection of videos (as blobs) within a container

C#

```
// Runs the Content Moderator Job on all Blobs in a given container name
static void RunContentModeratorJobOnBlobs()
{
    // Get the reference to the list of Blobs. See the following method.
    var blobList = GetBlobsList();

    // Iterate over the Blob list items or work on specific ones as needed
    foreach (var sourceBlob in blobList)
    {
        // Create an Asset
        IAsset asset =
_context.Assets.CreateFromBlob((CloudBlockBlob)sourceBlob,
                            _StorageCredentials, AssetCreationOptions.None);
        asset.Update();

        // Submit to Content Moderator
        RunContentModeratorJob(asset);
    }
}

// Get all blobs in your container
static IEnumerable<IListBlobItem> GetBlobsList()
{
    // Get a reference to the Container within the Storage Account
    // that has the files (blobs) for moderation
    CloudBlobClient CloudBlobClient =
_StorageAccount.CreateCloudBlobClient();
    CloudBlobContainer MediaBlobContainer =
CloudBlobClient.GetContainerReference(STORAGE_CONTAINER_NAME);

    // Get the reference to the list of Blobs
    var blobList = MediaBlobContainer.ListBlobs();
    return blobList;
}
```

Add the method to run the Content Moderator Job

C#

```
// Run the Content Moderator job on the designated Asset from local file or
blob storage
```

```
static void RunContentModeratorJob(IAsset asset)
{
    // Grab the presets
    string configuration = File.ReadAllText(CONTENT_MODERATOR_PRESET_FILE);

    // grab instance of Azure Media Content Moderator MP
    IMediaProcessor mp =
_context.MediaProcessors.GetLatestMediaProcessorByName(MEDIA_PROCESSOR);

    // create Job with Content Moderator task
    IJob job = _context.Jobs.Create(String.Format("Content Moderator {0}",
            asset.AssetFiles.First() + "_" + Guid.NewGuid()));

    ITask contentModeratorTask = job.Tasks.AddNew("Adult and racy classifier
task",
            mp, configuration,
            TaskOptions.None);
    contentModeratorTask.InputAssets.Add(asset);
    contentModeratorTask.OutputAssets.AddNew("Adult and racy classifier
output",
        AssetCreationOptions.None);

    job.Submit();


    // Create progress printing and querying tasks
    Task progressPrintTask = new Task(() =>
    {
        IJob jobQuery = null;
        do
        {
            var progressContext = _context;
            jobQuery = progressContext.Jobs
            .Where(j => j.Id == job.Id)
                .First();
                Console.WriteLine(string.Format("{0}\t{1}",
                DateTime.Now,
                jobQuery.State));
                Thread.Sleep(10000);
            }
            while (jobQuery.State != JobState.Finished &&
            jobQuery.State != JobState.Error &&
            jobQuery.State != JobState.Canceled);
    });
    progressPrintTask.Start();

    Task progressJobTask = job.GetExecutionProgressTask(
    CancellationToken.None);
    progressJobTask.Wait();

    // If job state is Error, the event handling
    // method for job progress should log errors.  Here we check
    // for error state and exit if needed.
    if (job.State == JobState.Error)
    {
```

```
        ErrorDetail error = job.Tasks.First().ErrorDetails.First();
        Console.WriteLine(string.Format("Error: {0}. {1}",
        error.Code,
        error.Message));
    }

    DownloadAsset(job.OutputMediaAssets.First(), OUTPUT_FOLDER);
}
```

Add helper functions

These methods download the Content Moderator output file (JSON) from the Azure Media Services asset, and help track the status of the moderation job so that the program can log a running status to the console.

C#

```
static void DownloadAsset(IAsset asset, string outputDirectory)
{
    foreach (IAssetFile file in asset.AssetFiles)
    {
        file.Download(Path.Combine(outputDirectory, file.Name));
    }
}

// event handler for Job State
static void StateChanged(object sender, JobStateChangedEventArgs e)
{
    Console.WriteLine("Job state changed event:");
    Console.WriteLine("  Previous state: " + e.PreviousState);
    Console.WriteLine("  Current state: " + e.CurrentState);
    switch (e.CurrentState)
    {
        case JobState.Finished:
            Console.WriteLine();
            Console.WriteLine("Job finished.");
            break;
        case JobState.Canceling:
        case JobState.Queued:
        case JobState.Scheduled:
        case JobState.Processing:
            Console.WriteLine("Please wait...\n");
            break;
        case JobState.Canceled:
            Console.WriteLine("Job is canceled.\n");
            break;
        case JobState.Error:
            Console.WriteLine("Job failed.\n");
            break;
        default:
            break;
```

```
    }
}
```

Run the program and review the output

After the Content Moderation job is completed, analyze the JSON response. It consists of these elements:

- Video information summary
- **Shots** as "**fragments**"
- **Key frames** as "**events**" with a **reviewRecommended" (= true or false)"** flag based on **Adult** and **Racy** scores
- **start**, **duration**, **totalDuration**, and **timestamp** are in "ticks". Divide by **timescale** to get the number in seconds.

> **Note**
>
> - `adultScore` represents the potential presence and prediction score of content that may be considered sexually explicit or adult in certain situations.
> - `racyScore` represents the potential presence and prediction score of content that may be considered sexually suggestive or mature in certain situations.
> - `adultScore` and `racyScore` are between 0 and 1. The higher the score, the higher the model is predicting that the category may be applicable. This preview relies on a statistical model rather than manually coded outcomes. We recommend testing with your own content to determine how each category aligns to your requirements.
> - `reviewRecommended` is either true or false depending on the internal score thresholds. Customers should assess whether to use this value or decide on custom thresholds based on their content policies.

JSON


```
{
"version": 2,
"timescale": 90000,
"offset": 0,
"framerate": 50,
"width": 1280,
"height": 720,
"totalDuration": 18696321,
"fragments": [
{
```

```
    "start": 0,
    "duration": 18000
},
{
    "start": 18000,
    "duration": 3600,
    "interval": 3600,
    "events": [
    [
        {
        "reviewRecommended": false,
        "adultScore": 0.00001,
        "racyScore": 0.03077,
        "index": 5,
        "timestamp": 18000,
        "shotIndex": 0
        }
    ]
    ]
},
{
    "start": 18386372,
    "duration": 119149,
    "interval": 119149,
    "events": [
    [
        {
        "reviewRecommended": true,
        "adultScore": 0.00000,
        "racyScore": 0.91902,
        "index": 5085,
        "timestamp": 18386372,
        "shotIndex": 62
        }
    ]
    ]
}
]
}
```


Next steps

Download the Visual Studio solution for this and other Content Moderator quickstarts for .NET.

Moderate with custom image lists in C#

Article • 01/18/2024

This article provides information and code samples to help you get started using the Content Moderator SDK for .NET to:

- Create a custom image list
- Add and remove images from the list
- Get the IDs of all images in the list
- Retrieve and update list metadata
- Refresh the list search index
- Screen images against images in the list
- Delete all images from the list
- Delete the custom list

> **Note**
>
> There is a maximum limit of **5 image lists** with each list **not to exceed 10,000 images**.

The console application for this guide simulates some of the tasks you can perform with the image list API.

If you don't have an Azure subscription, create a free account before you begin.

Sign up for Content Moderator services

Before you can use Content Moderator services through the REST API or the SDK, you'll need an API subscription key. Subscribe to Content Moderator service in the Azure portal to obtain it.

Create your Visual Studio project

1. Add a new **Console app (.NET Framework)** project to your solution.

   In the sample code, name the project **ImageLists**.

2. Select this project as the single startup project for the solution.

Install required packages

Install the following NuGet packages:

- Microsoft.Azure.CognitiveServices.ContentModerator
- Microsoft.Rest.ClientRuntime
- Newtonsoft.Json

Update the program's using statements

Add the following `using` statements

C#

```
using Microsoft.Azure.CognitiveServices.ContentModerator;
using Microsoft.Azure.CognitiveServices.ContentModerator.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
```

Create the Content Moderator client

Add the following code to create a Content Moderator client for your subscription. Update the `AzureEndpoint` and `CMSubscriptionKey` fields with the values of your endpoint URL and subscription key. You can find these in the **Quick start** tab of your resource in the Azure portal.

C#

```
/// <summary>
/// Wraps the creation and configuration of a Content Moderator client.
/// </summary>
/// <remarks>This class library contains insecure code. If you adapt this
/// code for use in production, use a secure method of storing and using
/// your Content Moderator subscription key.</remarks>
public static class Clients
{
    /// <summary>
    /// The base URL for Content Moderator calls.
    /// </summary>
    private static readonly string AzureEndpoint = "YOUR ENDPOINT URL";

    /// <summary>
    /// Your Content Moderator subscription key.
    /// </summary>
    private static readonly string CMSubscriptionKey = "YOUR API KEY";
```

```
    /// <summary>
    /// Returns a new Content Moderator client for your subscription.
    /// </summary>
    /// <returns>The new client.</returns>
    /// <remarks>The <see cref="ContentModeratorClient"/> is disposable.
    /// When you have finished using the client,
    /// you should dispose of it either directly or indirectly. </remarks>
    public static ContentModeratorClient NewClient()
    {
        // Create and initialize an instance of the Content Moderator API
wrapper.
        ContentModeratorClient client = new ContentModeratorClient(new
ApiKeyServiceClientCredentials(CMSubscriptionKey));

        client.Endpoint = AzureEndpoint;
        return client;
    }
}
```

> **ⓘ Important**
>
> Remember to remove the key from your code when you're done, and never post it publicly. For production, use a secure way of storing and accessing your credentials like **Azure Key Vault**. See the Azure AI services **security** article for more information.

Initialize application-specific settings

Add the following classes and static fields to the **Program** class in Program.cs.

C#

```
/// <summary>
/// The minimum amount of time, im milliseconds, to wait between calls
/// to the Image List API.
/// </summary>
private const int throttleRate = 3000;

/// <summary>
/// The number of minutes to delay after updating the search index before
/// performing image match operations against the list.
/// </summary>
private const double latencyDelay = 0.5;

/// <summary>
/// Define constants for the labels to apply to the image list.
/// </summary>
private class Labels
{
    public const string Sports = "Sports";
```

```
    public const string Swimsuit = "Swimsuit";
}

/// <summary>
/// Define input data for images for this sample.
/// </summary>
private class Images
{
    /// <summary>
    /// Represents a group of images that all share the same label.
    /// </summary>
    public class Data
    {
        /// <summary>
        /// The label for the images.
        /// </summary>
        public string Label;

        /// <summary>
        /// The URLs of the images.
        /// </summary>
        public string[] Urls;
    }

    /// <summary>
    /// The initial set of images to add to the list with the sports label.
    /// </summary>
    public static readonly Data Sports = new Data()
    {
        Label = Labels.Sports,
        Urls = new string[] {

"https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample6.png",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample9.png"
        }
    };

    /// <summary>
    /// The initial set of images to add to the list with the swimsuit
label.
    /// </summary>
    /// <remarks>We're adding sample16.png (image of a puppy), to simulate
    /// an improperly added image that we will later remove from the list.
    /// Note: each image can have only one entry in a list, so sample4.png
    /// will throw an exception when we try to add it with a new label.
</remarks>
    public static readonly Data Swimsuit = new Data()
    {
        Label = Labels.Swimsuit,
        Urls = new string[] {

"https://moderatorsampleimages.blob.core.windows.net/samples/sample1.jpg",
```

```
"https://moderatorsampleimages.blob.core.windows.net/samples/sample3.png",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png"
        }
    };

    /// <summary>
    /// The set of images to subsequently remove from the list.
    /// </summary>
    public static readonly string[] Corrections = new string[] {

"https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png"
    };
}

/// <summary>
/// The images to match against the image list.
/// </summary>
/// <remarks>Samples 1 and 4 should scan as matches; samples 5 and 16 should
not.</remarks>
private static readonly string[] ImagesToScreen = new string[] {

"https://moderatorsampleimages.blob.core.windows.net/samples/sample1.jpg",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample5.png",

"https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png"
};

/// <summary>
/// A dictionary that tracks the ID assigned to each image URL when
/// the image is added to the list.
/// </summary>
/// <remarks>Indexed by URL.</remarks>
private static readonly Dictionary<string, int> ImageIdMap =
    new Dictionary<string, int>();

/// <summary>
/// The name of the file to contain the output from the list management
operations.
/// </summary>
/// <remarks>Relative paths are relative to the execution directory.
</remarks>
private static string OutputFile = "ListOutput.log";

/// <summary>
/// A static reference to the text writer to use for logging.
/// </summary>
private static TextWriter writer;
```

```
/// <summary>
/// A copy of the list details.
/// </summary>
/// <remarks>Used to initially create the list, and later to update the
/// list details.</remarks>
private static Body listDetails;
```

> **Note**
>
> Your Content Moderator service key has a requests-per-second (RPS) rate limit, and if you exceed the limit, the SDK throws an exception with a 429 error code. A free tier key has a one-RPS rate limit.

Create a method to write messages to the log file

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Writes a message to the log file, and optionally to the console.
/// </summary>
/// <param name="message">The message.</param>
/// <param name="echo">if set to <c>true</c>, write the message to the
console.</param>
private static void WriteLine(string message = null, bool echo = false)
{
    writer.WriteLine(message ?? String.Empty);

    if (echo)
    {
        Console.WriteLine(message ?? String.Empty);
    }
}
```

Create a method to create the custom list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Creates the custom list.
```

```
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <returns>The response object from the operation.</returns>
private static ImageList CreateCustomList(ContentModeratorClient client)
{
    // Create the request body.
    listDetails = new Body("MyList", "A sample list",
        new BodyMetadata("Acceptable", "Potentially racy"));

    WriteLine($"Creating list {listDetails.Name}.", true);

    var result = client.ListManagementImageLists.Create(
        "application/json", listDetails);
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));

    return result;
}
```

Create a method to add a collection of images to the list

Add the following method to the **Program** class. This guide does not demonstrate how to apply tags to images in the list.

C#

```
/// <summary>
/// Adds images to an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <param name="imagesToAdd">The images to add.</param>
/// <param name="label">The label to apply to each image.</param>
/// <remarks>Images are assigned content IDs when they are added to the
list.
/// Track the content ID assigned to each image.</remarks>
private static void AddImages(
ContentModeratorClient client, int listId,
IEnumerable<string> imagesToAdd, string label)
{
    foreach (var imageUrl in imagesToAdd)
    {
        WriteLine();
        WriteLine($"Adding {imageUrl} to list {listId} with label {label}.",
true);
        try
        {
```

```
            var result = client.ListManagementImage.AddImageUrlInput(
                listId.ToString(), "application/json", new BodyModel("URL",
imageUrl), null, label);

            ImageIdMap.Add(imageUrl, Int32.Parse(result.ContentId));

            WriteLine("Response:");
            WriteLine(JsonConvert.SerializeObject(result,
Formatting.Indented));
        }
        catch (Exception ex)
        {
            WriteLine($"Unable to add image to list. Caught
{ex.GetType().FullName}: {ex.Message}", true);
        }
        finally
        {
            Thread.Sleep(throttleRate);
        }
    }
}
```

Create a method to remove images from the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Removes images from an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <param name="imagesToRemove">The images to remove.</param>
/// <remarks>Images are assigned content IDs when they are added to the
list.
/// Use the content ID to remove the image.</remarks>
private static void RemoveImages(
    ContentModeratorClient client, int listId,
    IEnumerable<string> imagesToRemove)
{
    foreach (var imageUrl in imagesToRemove)
    {
        if (!ImageIdMap.ContainsKey(imageUrl)) continue;
        int imageId = ImageIdMap[imageUrl];

        WriteLine();
        WriteLine($"Removing entry for {imageUrl} (ID = {imageId}) from list
{listId}.", true);
```

```
            var result = client.ListManagementImage.DeleteImage(
                listId.ToString(), imageId.ToString());
            Thread.Sleep(throttleRate);

            ImageIdMap.Remove(imageUrl);

            WriteLine("Response:");
            WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));
        }
    }
```

Create a method to get all of the content IDs for images in the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Gets all image IDs in an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <returns>The response object from the operation.</returns>
private static ImageIds GetAllImageIds(
    ContentModeratorClient client, int listId)
{
    WriteLine();
    WriteLine($"Getting all image IDs for list {listId}.", true);

    var result =
client.ListManagementImage.GetAllImageIds(listId.ToString());
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));

    return result;
}
```

Create a method to update the details of the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Updates the details of an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <returns>The response object from the operation.</returns>
private static ImageList UpdateListDetails(
    ContentModeratorClient client, int listId)
{
    WriteLine();
    WriteLine($"Updating details for list {listId}.", true);

    listDetails.Name = "Swimsuits and sports";

    var result = client.ListManagementImageLists.Update(
        listId.ToString(), "application/json", listDetails);
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));

    return result;
}
```

Create a method to retrieve the details of the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Gets the details for an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <returns>The response object from the operation.</returns>
private static ImageList GetListDetails(
    ContentModeratorClient client, int listId)
{
    WriteLine();
    WriteLine($"Getting details for list {listId}.", true);

    var result =
client.ListManagementImageLists.GetDetails(listId.ToString());
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));
```

```
    return result;
}
```

Create a method to refresh the search index of the list

Add the following method to the **Program** class. Any time you update a list, you need to refresh the search index before using the list to screen images.

C#

```
/// <summary>
/// Refreshes the search index for an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <returns>The response object from the operation.</returns>
private static RefreshIndex RefreshSearchIndex(
    ContentModeratorClient client, int listId)
{
    WriteLine();
    WriteLine($"Refreshing the search index for list {listId}.", true);

    var result =
client.ListManagementImageLists.RefreshIndexMethod(listId.ToString());
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));

    return result;
}
```

Create a method to match images against the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Matches images against an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
/// <param name="imagesToMatch">The images to screen.</param>
```

```
private static void MatchImages(
    ContentModeratorClient client, int listId,
    IEnumerable<string> imagesToMatch)
{
    foreach (var imageUrl in imagesToMatch)
    {
        WriteLine();
        WriteLine($"Matching image {imageUrl} against list {listId}.",
true);

        var result = client.ImageModeration.MatchUrlInput(
                "application/json", new BodyModel("URL", imageUrl),
listId.ToString());
        Thread.Sleep(throttleRate);

        WriteLine("Response:");
        WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));
    }
}
```

Create a method to delete all images from the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Deletes all images from an image list.
/// </summary>
/// <param name="client">The Content Modertor client.</param>
/// <param name="listId">The list identifier.</param>
private static void DeleteAllImages(
    ContentModeratorClient client, int listId)
{
    WriteLine();
    WriteLine($"Deleting all images from list {listId}.", true);

    var result =
client.ListManagementImage.DeleteAllImages(listId.ToString());
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));
}
```

Create a method to delete the list

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Deletes an image list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="listId">The list identifier.</param>
private static void DeleteCustomList(
    ContentModeratorClient client, int listId)
{
    WriteLine();
    WriteLine($"Deleting list {listId}.", true);

    var result = client.ListManagementImageLists.Delete(listId.ToString());
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));
}
```

Create a method to retrieve IDs for all image lists

Add the following method to the **Program** class.

C#

```
/// <summary>
/// Gets all list identifiers for the client.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <returns>The response object from the operation.</returns>
private static IList<ImageList> GetAllListIds(ContentModeratorClient client)
{
    WriteLine();
    WriteLine($"Getting all image list IDs.", true);

    var result = client.ListManagementImageLists.GetAllImageLists();
    Thread.Sleep(throttleRate);

    WriteLine("Response:");
    WriteLine(JsonConvert.SerializeObject(result, Formatting.Indented));

    return result;
}
```

Add code to simulate the use of an image list

Add the following code to the **Main** method. This code simulates many of the operations that you would perform in defining and managing the list, as well as using the list to screen images. The logging features allow you to see the response objects generated by the SDK calls to the Content Moderator service.

C#

```
// Create the text writer to use for logging, and cache a static reference
to it.
using (StreamWriter outputWriter = new StreamWriter(OutputFile))
{
    writer = outputWriter;

    // Create a Content Moderator client.
    using (var client = Clients.NewClient())
    {
        // Create a custom image list and record the ID assigned to it.
        var creationResult = CreateCustomList(client);
        if (creationResult.Id.HasValue)
        {
            // Cache the ID of the new image list.
            int listId = creationResult.Id.Value;

            // Perform various operations using the image list.
            AddImages(client, listId, Images.Sports.Urls,
Images.Sports.Label);
            AddImages(client, listId, Images.Swimsuit.Urls,
Images.Swimsuit.Label);

            GetAllImageIds(client, listId);
            UpdateListDetails(client, listId);
            GetListDetails(client, listId);

            // Be sure to refresh search index
            RefreshSearchIndex(client, listId);

            // WriteLine();
            WriteLine($"Waiting {latencyDelay} minutes to allow the server
time to propagate the index changes.", true);
            Thread.Sleep((int)(latencyDelay * 60 * 1000));

            // Match images against the image list.
            MatchImages(client, listId, ImagesToMatch);

            // Remove images
            RemoveImages(client, listId, Images.Corrections);

            // Be sure to refresh search index
            RefreshSearchIndex(client, listId);
```

```
            WriteLine();
            WriteLine($"Waiting {latencyDelay} minutes to allow the server
time to propagate the index changes.", true);
            Thread.Sleep((int)(latencyDelay * 60 * 1000));

            // Match images again against the image list. The removed image
should not get matched.
            MatchImages(client, listId, ImagesToMatch);

            // Delete all images from the list.
            DeleteAllImages(client, listId);

            // Delete the image list.
            DeleteCustomList(client, listId);

            // Verify that the list was deleted.
            GetAllListIds(client);
            }
        }

        writer.Flush();
        writer.Close();
        writer = null;
}

Console.WriteLine();
Console.WriteLine("Press any key to exit...");
Console.ReadKey();
```

Run the program and review the output

The list ID and the image content IDs are different each time you run the application. The log file written by the program has the following output:

JSON

```
Creating list MyList.
Response:
{
    "Id": 169642,
    "Name": "MyList",
    "Description": "A sample list",
    "Metadata": {
        "Key One": "Acceptable",
        "Key Two": "Potentially racy"
    }
}

Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png to
list 169642 with label Sports.
```

```
Response:
{
    "ContentId": "169490",
    "AdditionalInfo": [
    {
        "Key": "Source",
        "Value": "169642"
    },
    {
        "Key": "ImageDownloadTimeInMs",
        "Value": "233"
    },
    {
        "Key": "ImageSizeInBytes",
        "Value": "2945548"
    }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_b4d3e20a-0751-
4760-8829-475e5da33ce8"
}

Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample6.png to
list 169642 with label Sports.
Response:
{
    "ContentId": "169491",
    "AdditionalInfo": [
    {
        "Key": "Source",
        "Value": "169642"
    },
    {
        "Key": "ImageDownloadTimeInMs",
        "Value": "215"
    },
    {
        "Key": "ImageSizeInBytes",
        "Value": "2440050"
    }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_cc1eb6af-2463-
4e5e-9145-2a11dcecbc30"
```

```
}

Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample9.png to
list 169642 with label Sports.
Response:
{
    "ContentId": "169492",
    "AdditionalInfo": [
    {
        "Key": "Source",
        "Value": "169642"
    },
    {
        "Key": "ImageDownloadTimeInMs",
        "Value": "98"
    },
    {
        "Key": "ImageSizeInBytes",
        "Value": "1631958"
    }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_01edc1f2-b448-
48cf-b7f6-23b64d5040e9"
}

Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample1.jpg to
list 169642 with label Swimsuit.
Response:
{
    "ContentId": "169493",
    "AdditionalInfo": [
    {
        "Key": "Source",
        "Value": "169642"
    },
    {
        "Key": "ImageDownloadTimeInMs",
        "Value": "27"
    },
    {
        "Key": "ImageSizeInBytes",
        "Value": "17280"
    }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
```

```
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_41f7bc6f-8778-
4576-ba46-37b43a6c2434"
}
```


```
Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample3.png to
list 169642 with label Swimsuit.
Response:
```


```
{
    "ContentId": "169494",
    "AdditionalInfo": [
    {
        "Key": "Source",
        "Value": "169642"
    },
    {
        "Key": "ImageDownloadTimeInMs",
        "Value": "129"
    },
    {
        "Key": "ImageSizeInBytes",
        "Value": "1242855"
    }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_61a48f33-eb55-
4fd9-ac97-20eb0f3622a5"
}
```


```
Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png to
list 169642 with label Swimsuit.
Unable to add image to list. Caught
Microsoft.CognitiveServices.ContentModerator.Models.APIErrorException:
Operation returned an invalid status code 'Conflict'

Adding
https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png to
list 169642 with label Swimsuit.
Response:
```


```
{
    "ContentId": "169495",
    "AdditionalInfo": [
    {
        "Key": "Source",
        "Value": "169642"
    },
```

```
    {
        "Key": "ImageDownloadTimeInMs",
        "Value": "65"
    },
    {
        "Key": "ImageSizeInBytes",
        "Value": "1088127"
    }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_1c1f3de4-58b9-
4aa8-82fa-1b0f479f6d7c"
}

Getting all image IDs for list 169642.
Response:
{
    "ContentSource": "169642",
    "ContentIds": [
        169490,
        169491,
        169492,
        169493,
        169494,
        169495
],
"Status": {
    "Code": 3000,
    "Description": "OK",
    "Exception": null
    },
"TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_0d017deb-38fa-
4701-a7b1-5b6608c79da2"
}

Updating details for list 169642.
Response:
{
    "Id": 169642,
    "Name": "Swimsuits and sports",
    "Description": "A sample list",
    "Metadata": {
        "Key One": "Acceptable",
        "Key Two": "Potentially racy"
    }
}

Getting details for list 169642.
Response:
```

```
{
    "Id": 169642,
    "Name": "Swimsuits and sports",
    "Description": "A sample list",
    "Metadata": {
        "Key One": "Acceptable",
        "Key Two": "Potentially racy"
    }
}

Refreshing the search index for list 169642.
Response:
{
    "ContentSourceId": "169642",
    "IsUpdateSuccess": true,
    "AdvancedInfo": [],
    "Status": {
        "Code": 3000,
        "Description": "RefreshIndex successfully completed.",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_c72255cd-55a0-
415e-9c18-0b9c08a9f25b"
}
Waiting 0.5 minutes to allow the server time to propagate the index changes.

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample1.jpg
against list 169642.
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_ec384878-dbaa-
4999-9042-6ac986355967",
    "CacheID": null,
    "IsMatch": true,
    "Matches": [
        {
            "Score": 1.0,
            "MatchId": 169493,
            "Source": "169642",
            "Tags": [],
            "Label": "Swimsuit"
        }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png
```

```
against list 169642.
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_e9db4b8f-3067-
400f-9552-d3e6af2474c0",
    "CacheID": null,
    "IsMatch": true,
    "Matches": [
        {
            "Score": 1.0,
            "MatchId": 169490,
            "Source": "169642",
            "Tags": [],
            "Label": "Sports"
        }
    ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
        }
}

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample5.png
against list 169642.
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_25991575-05da-
4904-89db-abe88270b403",
    "CacheID": null,
    "IsMatch": false,
    "Matches": [],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png
against list 169642.
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_c65d1c91-0d8a-
4511-8ac6-814e04adc845",
    "CacheID": null,
    "IsMatch": true,
    "Matches": [
        {
            "Score": 1.0,
```

```
            "MatchId": 169495,
            "Source": "169642",
            "Tags": [],
            "Label": "Swimsuit"
        }
        ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Removing entry for
https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png (ID
= 169495) from list 169642.
Response:
""

Refreshing the search index for list 169642.
Response:
{
    "ContentSourceId": "169642",
    "IsUpdateSuccess": true,
    "AdvancedInfo": [],
    "Status": {
        "Code": 3000,
        "Description": "RefreshIndex successfully completed.",
        "Exception": null
        },
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_b55a375e-30a1-
4612-aa7b-81edcee5bffb"
}

Waiting 0.5 minutes to allow the server time to propagate the index changes.

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample1.jpg
against list 169642.
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_00544948-2936-
489c-98c8-b507b654bff5",
    "CacheID": null,
    "IsMatch": true,
    "Matches": [
        {
            "Score": 1.0,
            "MatchId": 169493,
            "Source": "169642",
            "Tags": [],
            "Label": "Swimsuit"
        }
```

```
        ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample4.png
against list 169642.
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_c36ec646-53c2-
4705-86b2-d72b5c2273c7",
    "CacheID": null,
    "IsMatch": true,
    "Matches": [
        {
            "Score": 1.0,
            "MatchId": 169490,
            "Source": "169642",
            "Tags": [],
            "Label": "Sports"
        }
        ],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample5.png
against list 169642.
Response:
{
    TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_22edad74-690d-
4fbc-b7d0-bf64867c4cb9",
    "CacheID": null,
    "IsMatch": false,
    "Matches": [],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Matching image
https://moderatorsampleimages.blob.core.windows.net/samples/sample16.png
against list 169642.
```

```
Response:
{
    "TrackingId":
"WE_f0527c49616243c5ac65e1cc3482d390_ContentModerator.Preview_abd4a178-3238-
4601-8e4f-cf9ee66f605a",
    "CacheID": null,
    "IsMatch": false,
    "Matches": [],
    "Status": {
        "Code": 3000,
        "Description": "OK",
        "Exception": null
    }
}

Deleting all images from list 169642.
Response:
"Reset Successful."

Deleting list 169642.
Response:
""

Getting all image list IDs.
Response:
[]
```

Next steps

Get the Content Moderator .NET SDK and the Visual Studio solution for this and other Content Moderator quickstarts for .NET, and get started on your integration.

Check text against a custom term list in C#

Article • 01/18/2024

The default global list of terms in Azure Content Moderator is sufficient for most content moderation needs. However, you might need to screen for terms that are specific to your organization.

You can use the Content Moderator SDK for .NET to create custom lists of terms to use with the Text Moderation API.

This article provides information and code samples to help you get started using the Content Moderator SDK for .NET to:

- Create a list.
- Add terms to a list.
- Screen terms against the terms in a list.
- Delete terms from a list.
- Delete a list.
- Edit list information.
- Refresh the index so that changes to the list are included in a new scan.

If you don't have an Azure subscription, create a free account before you begin.

Sign up for Content Moderator services

Before you can use Content Moderator services through the REST API or the SDK, you'll need a subscription key. Subscribe to Content Moderator service in the Azure portal to obtain one.

Create your Visual Studio project

1. Add a new **Console app (.NET Framework)** project to your solution.
2. Name the project **TermLists**. Select this project as the single startup project for the solution.

Install required packages

Install the following NuGet packages for the TermLists project:

- Microsoft.Azure.CognitiveServices.ContentModerator
- Microsoft.Rest.ClientRuntime
- Microsoft.Rest.ClientRuntime.Azure
- Newtonsoft.Json

Update the program's using statements

Add the following `using` statements.

C#

```
using Microsoft.Azure.CognitiveServices.ContentModerator;
using Microsoft.Azure.CognitiveServices.ContentModerator.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
```

Create the Content Moderator client

Add the following code to create a Content Moderator client for your subscription. Update the `AzureEndpoint` and `CMSubscriptionKey` fields with the values of your endpoint URL and subscription key. You can find these in the **Quick start** tab of your resource in the Azure portal.

C#

```
/// <summary>
/// Wraps the creation and configuration of a Content Moderator client.
/// </summary>
/// <remarks>This class library contains insecure code. If you adapt this
/// code for use in production, use a secure method of storing and using
/// your Content Moderator subscription key.</remarks>
public static class Clients
{
    /// <summary>
    /// The base URL fragment for Content Moderator calls.
    /// </summary>
    private static readonly string AzureEndpoint = "YOUR ENDPOINT URL";

    /// <summary>
    /// Your Content Moderator subscription key.
    /// </summary>
    private static readonly string CMSubscriptionKey = "YOUR API KEY";

    /// <summary>
```

```
    /// Returns a new Content Moderator client for your subscription.
    /// </summary>
    /// <returns>The new client.</returns>
    /// <remarks>The <see cref="ContentModeratorClient"/> is disposable.
    /// When you have finished using the client,
    /// you should dispose of it either directly or indirectly. </remarks>
    public static ContentModeratorClient NewClient()
    {
        // Create and initialize an instance of the Content Moderator API
wrapper.
        ContentModeratorClient client = new ContentModeratorClient(new
ApiKeyServiceClientCredentials(CMSubscriptionKey));

        client.Endpoint = AzureEndpoint;
        return client;
    }
}
```

> **Important**
>
> Remember to remove the key from your code when you're done, and never post it publicly. For production, use a secure way of storing and accessing your credentials like **Azure Key Vault**. See the Azure AI services **security** article for more information.

Add private properties

Add the following private properties to namespace TermLists, class Program.

C#

```
/// <summary>
/// The language of the terms in the term lists.
/// </summary>
private const string lang = "eng";

/// <summary>
/// The minimum amount of time, in milliseconds, to wait between calls
/// to the Content Moderator APIs.
/// </summary>
private const int throttleRate = 3000;

/// <summary>
/// The number of minutes to delay after updating the search index before
/// performing image match operations against the list.
/// </summary>
private const double latencyDelay = 0.5;
```

Create a term list

You create a term list with **ContentModeratorClient.ListManagementTermLists.Create**. The first parameter to **Create** is a string that contains a MIME type, which should be "application/json". For more information, see the API reference. The second parameter is a **Body** object that contains a name and description for the new term list.

> **Note**
>
> There is a maximum limit of **5 term lists** with each list to **not exceed 10,000 terms**.

Add the following method definition to namespace TermLists, class Program.

> **Note**
>
> Your Content Moderator service key has a requests-per-second (RPS) rate limit, and if you exceed the limit, the SDK throws an exception with a 429 error code. A free tier key has a one-RPS rate limit.

C#

```
/// <summary>
/// Creates a new term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <returns>The term list ID.</returns>
static string CreateTermList (ContentModeratorClient client)
{
    Console.WriteLine("Creating term list.");

    Body body = new Body("Term list name", "Term list description");
    TermList list =
client.ListManagementTermLists.Create("application/json", body);
    if (false == list.Id.HasValue)
    {
        throw new Exception("TermList.Id value missing.");
    }
    else
    {
        string list_id = list.Id.Value.ToString();
        Console.WriteLine("Term list created. ID: {0}.", list_id);
        Thread.Sleep(throttleRate);
        return list_id;
    }
}
```

Update term list name and description

You update the term list information with **ContentModeratorClient.ListManagementTermLists.Update**. The first parameter to **Update** is the term list ID. The second parameter is a MIME type, which should be "application/json". For more information, see the API reference. The third parameter is a **Body** object, which contains the new name and description.

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Update the information for the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list to update.</param>
/// <param name="name">The new name for the term list.</param>
/// <param name="description">The new description for the term list.</param>
static void UpdateTermList (ContentModeratorClient client, string list_id,
string name = null, string description = null)
{
    Console.WriteLine("Updating information for term list with ID {0}.",
list_id);
    Body body = new Body(name, description);
    client.ListManagementTermLists.Update(list_id, "application/json",
body);
    Thread.Sleep(throttleRate);
}
```

Add a term to a term list

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Add a term to the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list to update.</param>
/// <param name="term">The term to add to the term list.</param>
static void AddTerm (ContentModeratorClient client, string list_id, string
term)
{
    Console.WriteLine("Adding term \"{0}\" to term list with ID {1}.", term,
list_id);
    client.ListManagementTerm.AddTerm(list_id, term, lang);
```

```
    Thread.Sleep(throttleRate);
}
```

Get all terms in a term list

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Get all terms in the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list from which to get all
terms.</param>
static void GetAllTerms(ContentModeratorClient client, string list_id)
{
    Console.WriteLine("Getting terms in term list with ID {0}.", list_id);
    Terms terms = client.ListManagementTerm.GetAllTerms(list_id, lang);
    TermsData data = terms.Data;
    foreach (TermsInList term in data.Terms)
    {
        Console.WriteLine(term.Term);
    }
    Thread.Sleep(throttleRate);
}
```

Add code to refresh the search index

After you make changes to a term list, you refresh its search index for the changes to be included the next time you use the term list to screen text. This is similar to how a search engine on your desktop (if enabled) or a web search engine continually refreshes its index to include new files or pages.

You refresh a term list search index with **ContentModeratorClient.ListManagementTermLists.RefreshIndexMethod**.

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Refresh the search index for the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list to refresh.</param>
static void RefreshSearchIndex (ContentModeratorClient client, string
```

```
list_id)
{
    Console.WriteLine("Refreshing search index for term list with ID {0}.",
list_id);
    client.ListManagementTermLists.RefreshIndexMethod(list_id, lang);
    Thread.Sleep((int)(latencyDelay * 60 * 1000));
}
```

Screen text using a term list

You screen text using a term list with **ContentModeratorClient.TextModeration.ScreenText**, which takes the following parameters.

- The language of the terms in the term list.
- A MIME type, which can be "text/html", "text/xml", "text/markdown", or "text/plain".
- The text to screen.
- A boolean value. Set this field to **true** to autocorrect the text before screening it.
- A boolean value. Set this field to **true** to detect personal data in the text.
- The term list ID.

For more information, see the API reference.

ScreenText returns a **Screen** object, which has a **Terms** property that lists any terms that Content Moderator detected in the screening. Note that if Content Moderator did not detect any terms during the screening, the **Terms** property has value **null**.

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Screen the indicated text for terms in the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list to use to screen the text.
</param>
/// <param name="text">The text to screen.</param>
static void ScreenText (ContentModeratorClient client, string list_id,
string text)
{
    Console.WriteLine("Screening text: \"{0}\" using term list with ID
{1}.", text, list_id);
    Screen screen = client.TextModeration.ScreenText(lang, "text/plain",
text, false, false, list_id);
    if (null == screen.Terms)
```

```
    {
        Console.WriteLine("No terms from the term list were detected in the
text.");
    }
    else
    {
        foreach (DetectedTerms term in screen.Terms)
        {
            Console.WriteLine(String.Format("Found term: \"{0}\" from list
ID {1} at index {2}.", term.Term, term.ListId, term.Index));
        }
    }
    Thread.Sleep(throttleRate);
}
```

Delete terms and lists

Deleting a term or a list is straightforward. You use the SDK to do the following tasks:

- Delete a term. (**ContentModeratorClient.ListManagementTerm.DeleteTerm**)
- Delete all the terms in a list without deleting the list. (**ContentModeratorClient.ListManagementTerm.DeleteAllTerms**)
- Delete a list and all of its contents. (**ContentModeratorClient.ListManagementTermLists.Delete**)

Delete a term

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Delete a term from the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list from which to delete the
term.</param>
/// <param name="term">The term to delete.</param>
static void DeleteTerm (ContentModeratorClient client, string list_id,
string term)
{
    Console.WriteLine("Removed term \"{0}\" from term list with ID {1}.",
term, list_id);
    client.ListManagementTerm.DeleteTerm(list_id, term, lang);
    Thread.Sleep(throttleRate);
}
```

Delete all terms in a term list

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Delete all terms from the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list from which to delete all
terms.</param>
static void DeleteAllTerms (ContentModeratorClient client, string list_id)
{
    Console.WriteLine("Removing all terms from term list with ID {0}.",
list_id);
    client.ListManagementTerm.DeleteAllTerms(list_id, lang);
    Thread.Sleep(throttleRate);
}
```

Delete a term list

Add the following method definition to namespace TermLists, class Program.

C#

```
/// <summary>
/// Delete the indicated term list.
/// </summary>
/// <param name="client">The Content Moderator client.</param>
/// <param name="list_id">The ID of the term list to delete.</param>
static void DeleteTermList (ContentModeratorClient client, string list_id)
{
    Console.WriteLine("Deleting term list with ID {0}.", list_id);
    client.ListManagementTermLists.Delete(list_id);
    Thread.Sleep(throttleRate);
}
```

Compose the Main method

Add the **Main** method definition to namespace **TermLists**, class **Program**. Finally, close the **Program** class and the **TermLists** namespace.

C#

```
static void Main(string[] args)
{
```

```
    using (var client = Clients.NewClient())
    {
        string list_id = CreateTermList(client);

        UpdateTermList(client, list_id, "name", "description");
        AddTerm(client, list_id, "term1");
        AddTerm(client, list_id, "term2");

        GetAllTerms(client, list_id);

        // Always remember to refresh the search index of your list
        RefreshSearchIndex(client, list_id);

        string text = "This text contains the terms \"term1\" and
\"term2\".";
        ScreenText(client, list_id, text);

        DeleteTerm(client, list_id, "term1");

        // Always remember to refresh the search index of your list
        RefreshSearchIndex(client, list_id);

        text = "This text contains the terms \"term1\" and \"term2\".";
        ScreenText(client, list_id, text);

        DeleteAllTerms(client, list_id);
        DeleteTermList(client, list_id);

        Console.WriteLine("Press ENTER to close the application.");
        Console.ReadLine();
    }
}
```

Run the application to see the output

Your console output will look like the following:

Console

```
Creating term list.
Term list created. ID: 252.
Updating information for term list with ID 252.

Adding term "term1" to term list with ID 252.
Adding term "term2" to term list with ID 252.

Getting terms in term list with ID 252.
term1
term2

Refreshing search index for term list with ID 252.
```

```
Screening text: "This text contains the terms "term1" and "term2"." using
term list with ID 252.
Found term: "term1" from list ID 252 at index 32.
Found term: "term2" from list ID 252 at index 46.

Removed term "term1" from term list with ID 252.

Refreshing search index for term list with ID 252.

Screening text: "This text contains the terms "term1" and "term2"." using
term list with ID 252.
Found term: "term2" from list ID 252 at index 46.

Removing all terms from term list with ID 252.
Deleting term list with ID 252.
Press ENTER to close the application.
```

Next steps

Get the Content Moderator .NET SDK and the Visual Studio solution for this and other Content Moderator quickstarts for .NET, and get started on your integration.

Export or delete user data in Content Moderator

Article • 01/18/2024

Important

Azure Content Moderator is being deprecated in February 2024, and will be retired by February 2027. It is being replaced by **Azure AI Content Safety**, which offers advanced AI features and enhanced performance.

Azure AI Content Safety is a comprehensive solution designed to detect harmful user-generated and AI-generated content in applications and services. Azure AI Content Safety is suitable for many scenarios such as online marketplaces, gaming companies, social messaging platforms, enterprise media companies, and K-12 education solution providers. Here's an overview of its features and capabilities:

- **Text and Image Detection APIs**: Scan text and images for sexual content, violence, hate, and self-harm with multiple severity levels.
- **Content Safety Studio**: An online tool designed to handle potentially offensive, risky, or undesirable content using our latest content moderation ML models. It provides templates and customized workflows that enable users to build their own content moderation systems.
- **Language support**: Azure AI Content Safety supports more than 100 languages and is specifically trained on English, German, Japanese, Spanish, French, Italian, Portuguese, and Chinese.

Azure AI Content Safety provides a robust and flexible solution for your content moderation needs. By switching from Content Moderator to Azure AI Content Safety, you can take advantage of the latest tools and technologies to ensure that your content is always moderated to your exact specifications.

Learn more about Azure AI Content Safety and explore how it can elevate your content moderation strategy.

Content Moderator collects user data to operate the service, but customers have full control to view, export, and delete their data using the Moderation APIs.

Note

This article provides steps about how to delete personal data from the device or service and can be used to support your obligations under the GDPR. For general information about GDPR, see the **GDPR section of the Microsoft Trust Center** and the **GDPR section of the Service Trust portal**.

For more information on how to export and delete user data in Content Moderator, see the following table.

Expand table

Data	Export Operation	Delete Operation
Account Info (Subscription Keys)	N/A	Delete using the Azure portal (Azure Subscriptions).
Images for custom matching	Call the Get image IDs API. Images are stored in a one-way proprietary hash format, and there is no way to extract the actual images.	Call the Delete all Images API. Or delete the Content Moderator resource using the Azure portal.
Terms for custom matching	Cal the Get all terms API	Call the Delete all terms API. Or delete the Content Moderator resource using the Azure portal.

Configure Azure AI services virtual networks

Article • 04/05/2024

Azure AI services provide a layered security model. This model enables you to secure your Azure AI services accounts to a specific subset of networks. When network rules are configured, only applications that request data over the specified set of networks can access the account. You can limit access to your resources with *request filtering*, which allows requests that originate only from specified IP addresses, IP ranges, or from a list of subnets in Azure Virtual Networks.

An application that accesses an Azure AI services resource when network rules are in effect requires authorization. Authorization is supported with Microsoft Entra ID credentials or with a valid API key.

> **Important**
>
> Turning on firewall rules for your Azure AI services account blocks incoming requests for data by default. To allow requests through, one of the following conditions needs to be met:
>
> - The request originates from a service that operates within an Azure Virtual Network on the allowed subnet list of the target Azure AI services account. The endpoint request that originated from the virtual network needs to be set as the **custom subdomain** of your Azure AI services account.
> - The request originates from an allowed list of IP addresses.
>
> Requests that are blocked include those from other Azure services, from the Azure portal, and from logging and metrics services.

> **Note**
>
> We recommend that you use the Azure Az PowerShell module to interact with Azure. See **Install Azure PowerShell** to get started. To learn how to migrate to the Az PowerShell module, see **Migrate Azure PowerShell from AzureRM to Az**.

Scenarios

To secure your Azure AI services resource, you should first configure a rule to deny access to traffic from all networks, including internet traffic, by default. Then, configure rules that grant access to traffic from specific virtual networks. This configuration enables you to build a secure network boundary for your applications. You can also configure rules to grant access to traffic from select public internet IP address ranges and enable connections from specific internet or on-premises clients.

Network rules are enforced on all network protocols to Azure AI services, including REST and WebSocket. To access data by using tools such as the Azure test consoles, explicit network rules must be configured. You can apply network rules to existing Azure AI services resources, or when you create new Azure AI services resources. After network rules are applied, they're enforced for all requests.

Supported regions and service offerings

Virtual networks are supported in regions where Azure AI services are available. Azure AI services support service tags for network rules configuration. The services listed here are included in the `CognitiveServicesManagement` service tag.

- ✓ Anomaly Detector
- ✓ Azure OpenAI
- ✓ Content Moderator
- ✓ Custom Vision
- ✓ Face
- ✓ Language Understanding (LUIS)
- ✓ Personalizer
- ✓ Speech service
- ✓ Language
- ✓ QnA Maker
- ✓ Translator

> **Note**
>
> If you use Azure OpenAI, LUIS, Speech Services, or Language services, the `CognitiveServicesManagement` tag only enables you to use the service by using the SDK or REST API. To access and use Azure OpenAI Studio, LUIS portal, Speech Studio, or Language Studio from a virtual network, you need to use the following tags:
>
> - `AzureActiveDirectory`
> - `AzureFrontDoor.Frontend`

- `AzureResourceManager`
- `CognitiveServicesManagement`
- `CognitiveServicesFrontEnd`
- `Storage` (Speech Studio only)

For information on configuring Azure AI Studio, see the Azure AI Studio documentation.

Change the default network access rule

By default, Azure AI services resources accept connections from clients on any network. To limit access to selected networks, you must first change the default action.

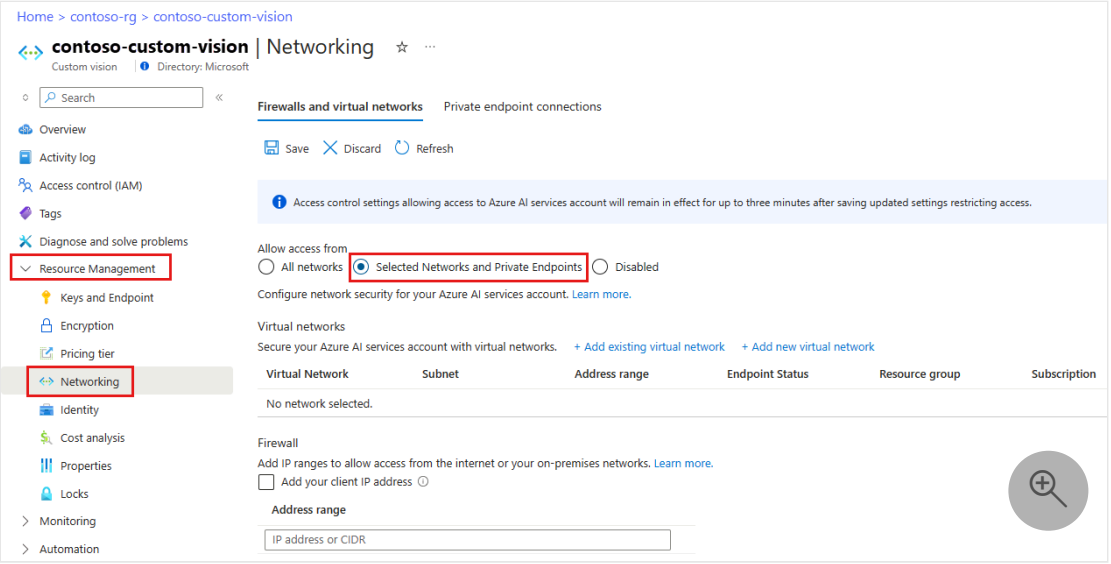
> ⚠ **Warning**
>
> Making changes to network rules can impact your applications' ability to connect to Azure AI services. Setting the default network rule to *deny* blocks all access to the data unless specific network rules that *grant* access are also applied.
>
> Before you change the default rule to deny access, be sure to grant access to any allowed networks by using network rules. If you allow listing for the IP addresses for your on-premises network, be sure to add all possible outgoing public IP addresses from your on-premises network.

Manage default network access rules

You can manage default network access rules for Azure AI services resources through the Azure portal, PowerShell, or the Azure CLI.

Azure portal

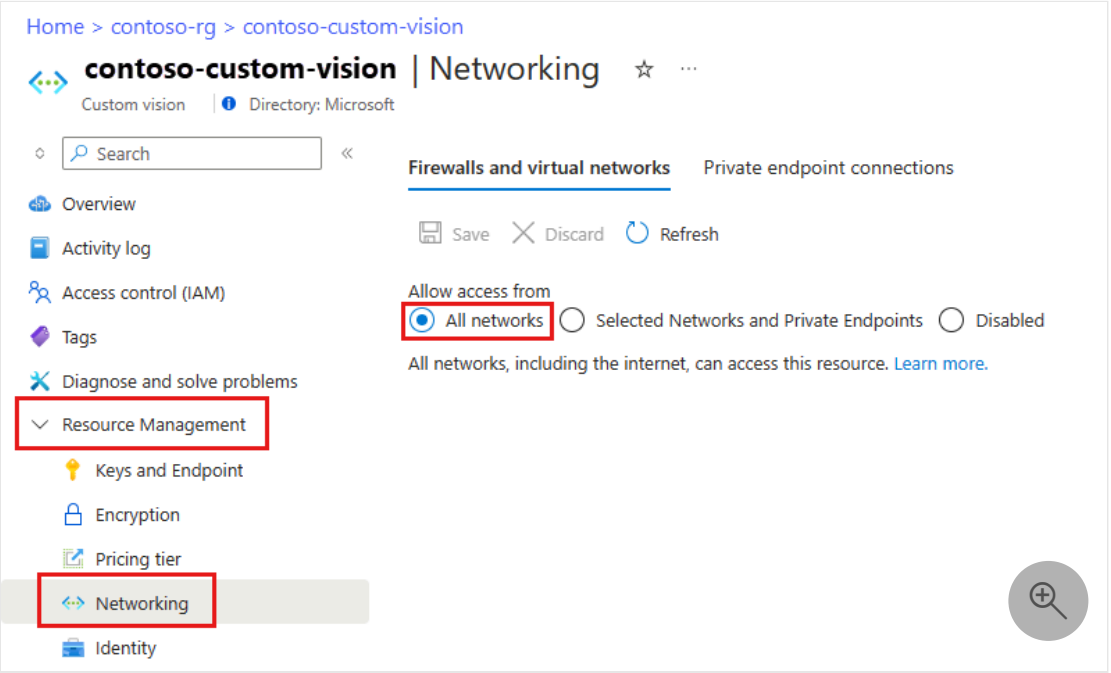
1. Go to the Azure AI services resource you want to secure.
2. Select **Resource Management** to expand it, then select **Networking**.



3. To deny access by default, under **Firewalls and virtual networks**, select **Selected Networks and Private Endpoints**.

   With this setting alone, unaccompanied by configured virtual networks or address ranges, all access is effectively denied. When all access is denied, requests that attempt to consume the Azure AI services resource aren't permitted. The Azure portal, Azure PowerShell, or the Azure CLI can still be used to configure the Azure AI services resource.

4. To allow traffic from all networks, select **All networks**.



5. Select **Save** to apply your changes.

Grant access from a virtual network

You can configure Azure AI services resources to allow access from specific subnets only. The allowed subnets might belong to a virtual network in the same subscription or in a different subscription. The other subscription can belong to a different Microsoft Entra tenant. When the subnet belongs to a different subscription, the Microsoft.CognitiveServices resource provider needs to be also registered for that subscription.

Enable a *service endpoint* for Azure AI services within the virtual network. The service endpoint routes traffic from the virtual network through an optimal path to the Azure AI service. For more information, see Virtual Network service endpoints.

The identities of the subnet and the virtual network are also transmitted with each request. Administrators can then configure network rules for the Azure AI services resource to allow requests from specific subnets in a virtual network. Clients granted access by these network rules must continue to meet the authorization requirements of the Azure AI services resource to access the data.

Each Azure AI services resource supports up to 100 virtual network rules, which can be combined with IP network rules. For more information, see Grant access from an internet IP range later in this article.

Set required permissions

To apply a virtual network rule to an Azure AI services resource, you need the appropriate permissions for the subnets to add. The required permission is the default *Contributor* role or the *Cognitive Services Contributor* role. Required permissions can also be added to custom role definitions.

The Azure AI services resource and the virtual networks that are granted access might be in different subscriptions, including subscriptions that are part of a different Microsoft Entra tenant.

> **Note**
>
> Configuration of rules that grant access to subnets in virtual networks that are a part of a different Microsoft Entra tenant are currently supported only through PowerShell, the Azure CLI, and the REST APIs. You can view these rules in the Azure portal, but you can't configure them.

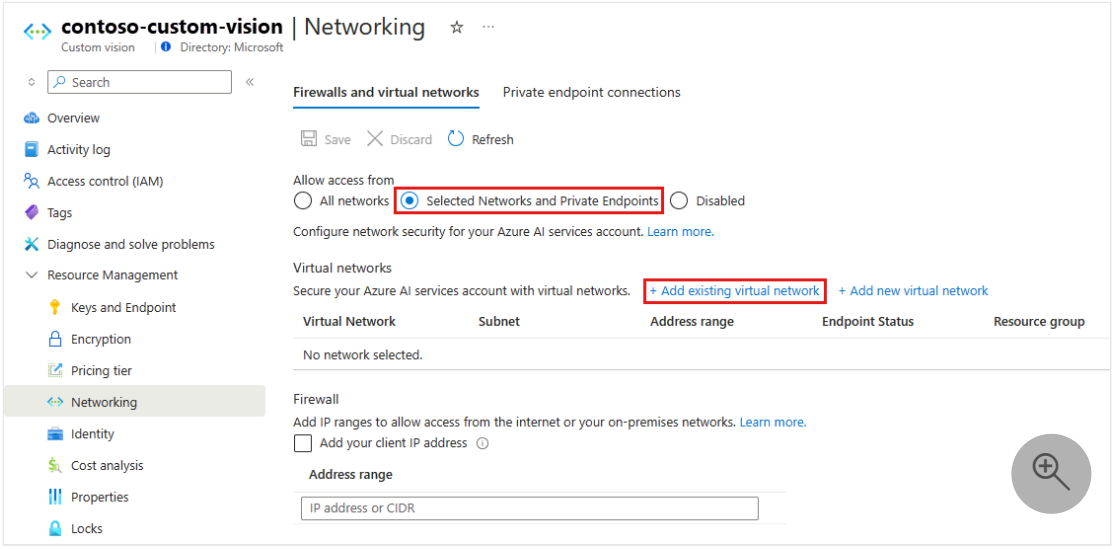
Configure virtual network rules

You can manage virtual network rules for Azure AI services resources through the Azure portal, PowerShell, or the Azure CLI.

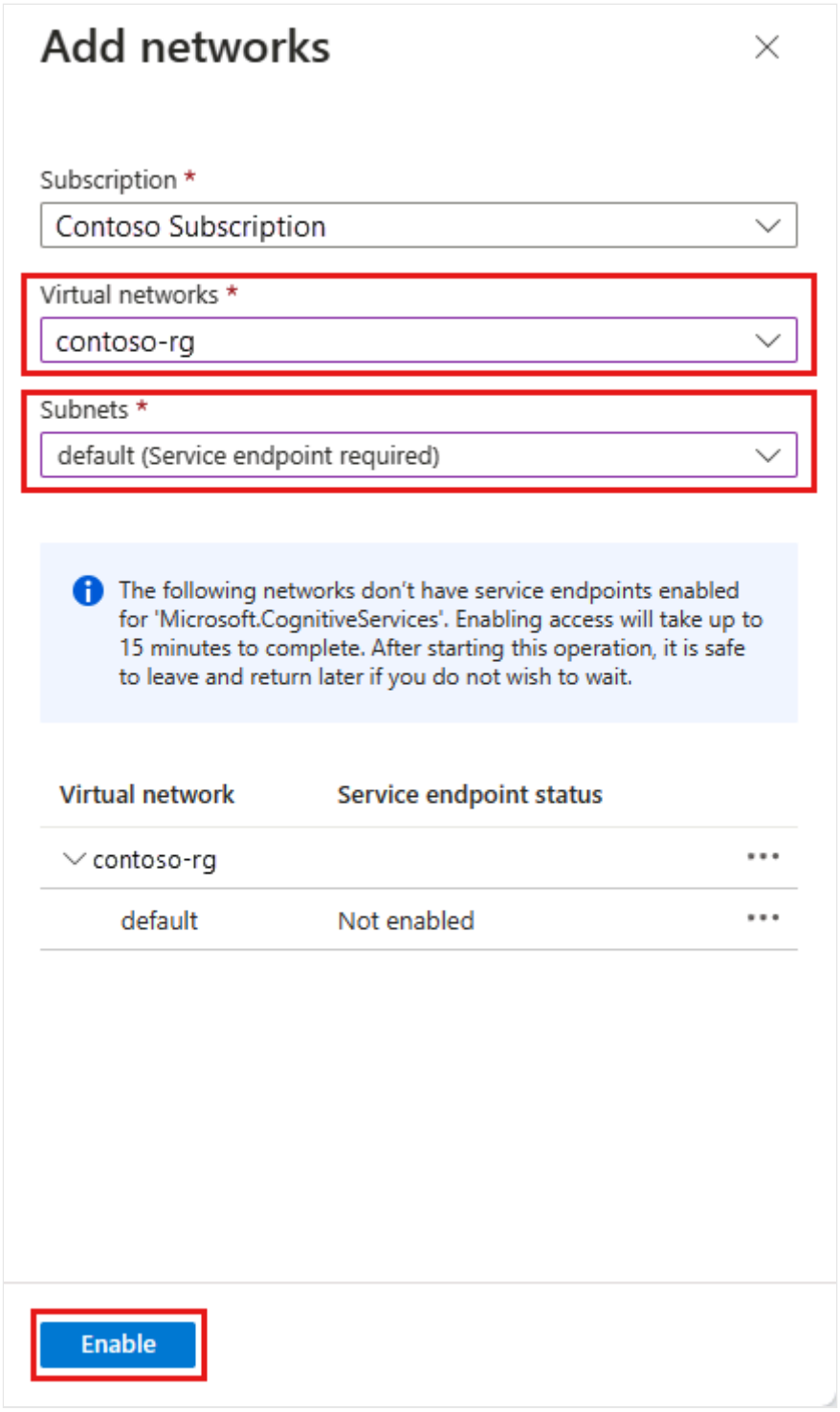
Azure portal

To grant access to a virtual network with an existing network rule:

1. Go to the Azure AI services resource you want to secure.
2. Select **Resource Management** to expand it, then select **Networking**.
3. Confirm that you selected **Selected Networks and Private Endpoints**.
4. Under **Allow access from**, select **Add existing virtual network**.



5. Select the **Virtual networks** and **Subnets** options, and then select **Enable**.

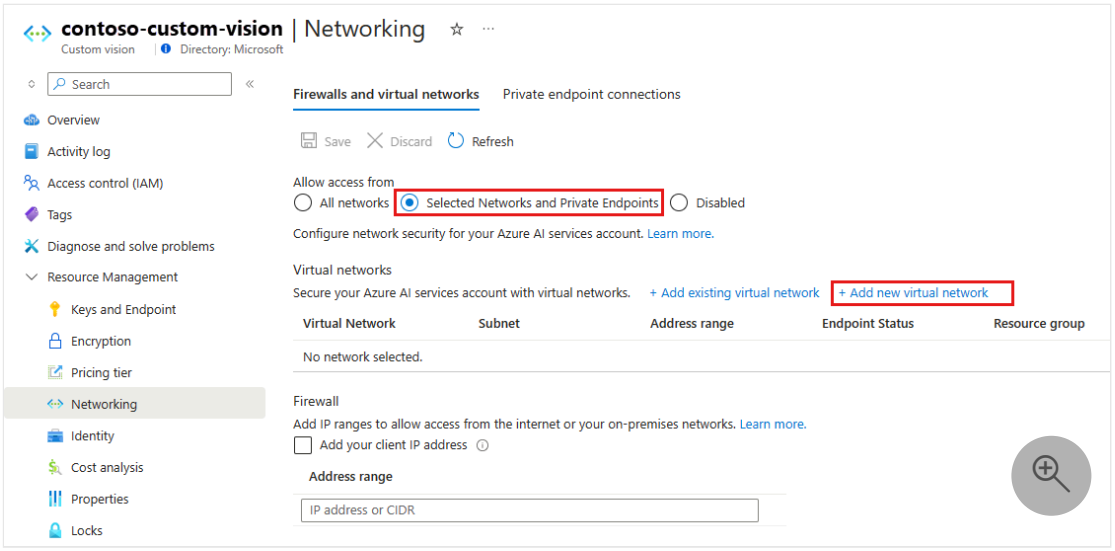


> **Note**
>
> If a service endpoint for Azure AI services wasn't previously configured for the selected virtual network and subnets, you can configure it as part of this operation.
>
> Currently, only virtual networks that belong to the same Microsoft Entra tenant are available for selection during rule creation. To grant access to a subnet in a virtual network that belongs to another tenant, use PowerShell, the Azure CLI, or the REST APIs.

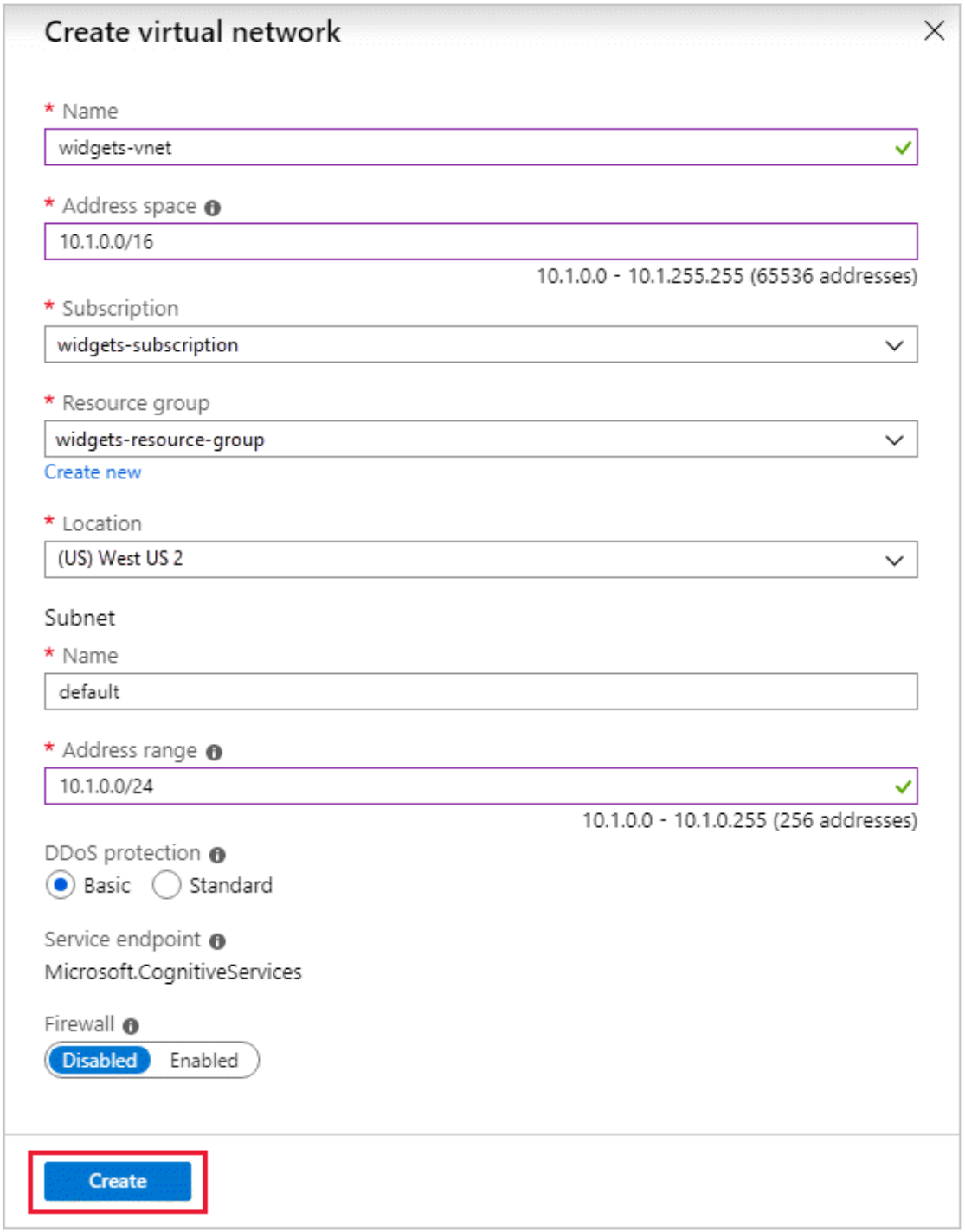
6. Select **Save** to apply your changes.

To create a new virtual network and grant it access:

1. On the same page as the previous procedure, select **Add new virtual network**.



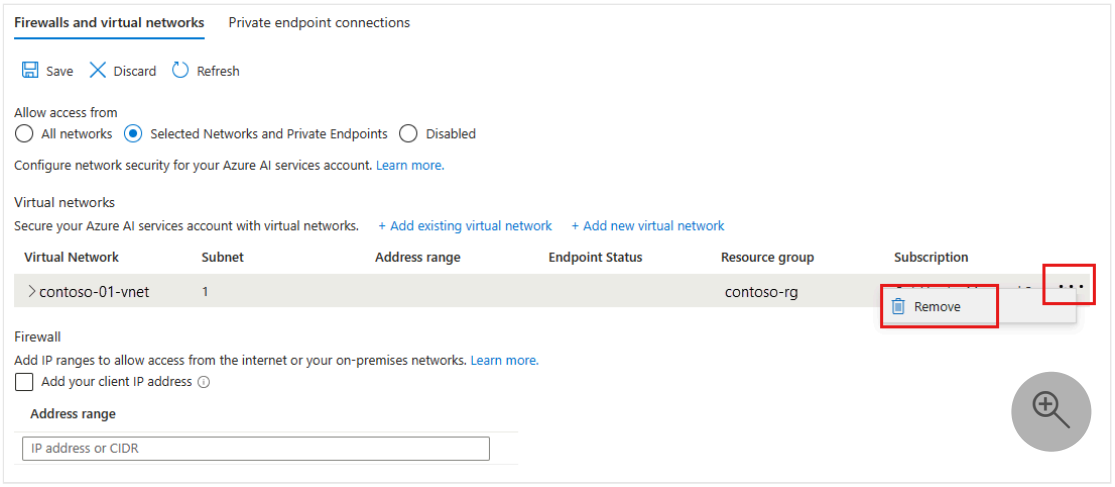
2. Provide the information necessary to create the new virtual network, and then select **Create**.



3. Select **Save** to apply your changes.

To remove a virtual network or subnet rule:

1. On the same page as the previous procedures, select **...(More options)** to open the context menu for the virtual network or subnet, and select **Remove**.



2. Select **Save** to apply your changes.

> **Important**
>
> Be sure to **set the default rule** to *deny*, or network rules have no effect.

Grant access from an internet IP range

You can configure Azure AI services resources to allow access from specific public internet IP address ranges. This configuration grants access to specific services and on-premises networks, which effectively block general internet traffic.

You can specify the allowed internet address ranges by using CIDR format (RFC 4632) in the form `192.168.0.0/16` or as individual IP addresses like `192.168.0.1`.

> **Tip**
>
> Small address ranges that use `/31` or `/32` prefix sizes aren't supported. Configure these ranges by using individual IP address rules.

IP network rules are only allowed for *public internet* IP addresses. IP address ranges reserved for private networks aren't allowed in IP rules. Private networks include addresses that start with `10.*`, `172.16.*` - `172.31.*`, and `192.168.*`. For more information, see Private Address Space (RFC 1918).

Currently, only IPv4 addresses are supported. Each Azure AI services resource supports up to 100 IP network rules, which can be combined with virtual network rules.

Configure access from on-premises networks

To grant access from your on-premises networks to your Azure AI services resource with an IP network rule, identify the internet-facing IP addresses used by your network. Contact your network administrator for help.

If you use Azure ExpressRoute on-premises for public peering or Microsoft peering, you need to identify the NAT IP addresses. For more information, see What is Azure ExpressRoute.

For public peering, each ExpressRoute circuit by default uses two NAT IP addresses. Each is applied to Azure service traffic when the traffic enters the Microsoft Azure network backbone. For Microsoft peering, the NAT IP addresses that are used are either customer provided or supplied by the service provider. To allow access to your service resources, you must allow these public IP addresses in the resource IP firewall setting.

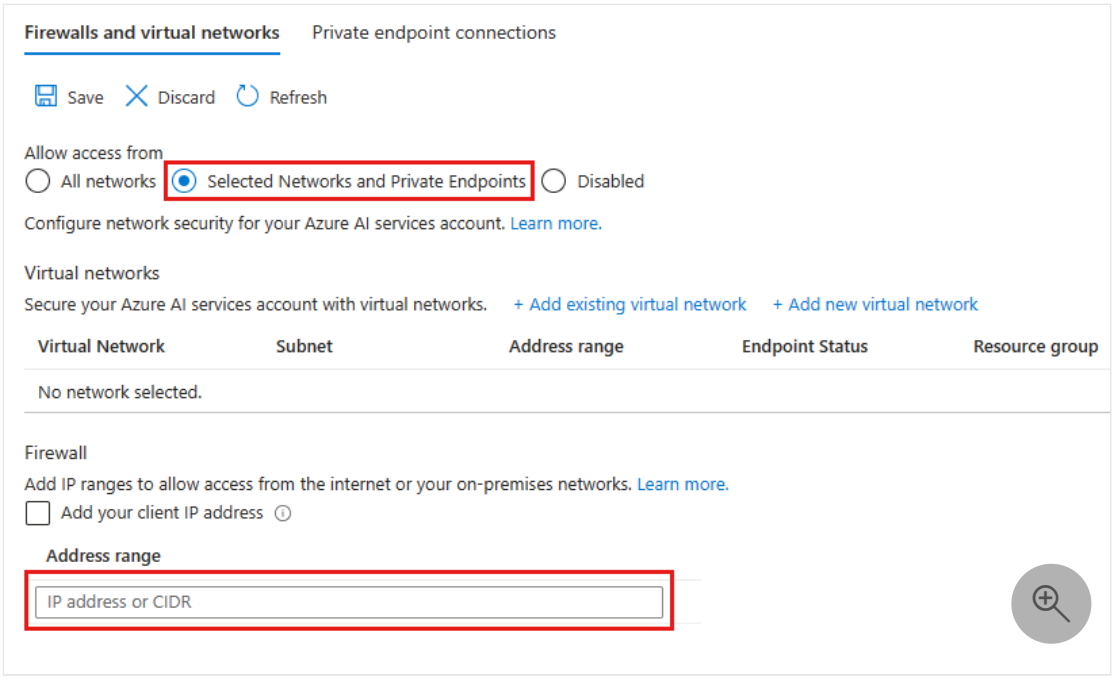
To find your public peering ExpressRoute circuit IP addresses, open a support ticket with ExpressRoute use the Azure portal. For more information, see NAT requirements for Azure public peering.

Managing IP network rules

You can manage IP network rules for Azure AI services resources through the Azure portal, PowerShell, or the Azure CLI.

Azure portal

1. Go to the Azure AI services resource you want to secure.
2. Select **Resource Management** to expand it, then select **Networking**.
3. Confirm that you selected **Selected Networks and Private Endpoints**.
4. Under **Firewalls and virtual networks**, locate the **Address range** option. To grant access to an internet IP range, enter the IP address or address range (in CIDR format). Only valid public IP (nonreserved) addresses are accepted.



To remove an IP network rule, select the trash can icon next to the address range.

5. Select **Save** to apply your changes.

> **Important**
>
> Be sure to **set the default rule** to *deny*, or network rules have no effect.

Use private endpoints

You can use private endpoints for your Azure AI services resources to allow clients on a virtual network to securely access data over Azure Private Link. The private endpoint uses an IP address from the virtual network address space for your Azure AI services resource. Network traffic between the clients on the virtual network and the resource traverses the virtual network and a private link on the Microsoft Azure backbone network, which eliminates exposure from the public internet.

Private endpoints for Azure AI services resources let you:

- Secure your Azure AI services resource by configuring the firewall to block all connections on the public endpoint for the Azure AI service.
- Increase security for the virtual network, by enabling you to block exfiltration of data from the virtual network.

- Securely connect to Azure AI services resources from on-premises networks that connect to the virtual network by using Azure VPN Gateway or ExpressRoutes with private-peering.

Understand private endpoints

A private endpoint is a special network interface for an Azure resource in your virtual network. Creating a private endpoint for your Azure AI services resource provides secure connectivity between clients in your virtual network and your resource. The private endpoint is assigned an IP address from the IP address range of your virtual network. The connection between the private endpoint and the Azure AI service uses a secure private link.

Applications in the virtual network can connect to the service over the private endpoint seamlessly. Connections use the same connection strings and authorization mechanisms that they would use otherwise. The exception is Speech Services, which require a separate endpoint. For more information, see Private endpoints with the Speech Services in this article. Private endpoints can be used with all protocols supported by the Azure AI services resource, including REST.

Private endpoints can be created in subnets that use service endpoints. Clients in a subnet can connect to one Azure AI services resource using private endpoint, while using service endpoints to access others. For more information, see Virtual Network service endpoints.

When you create a private endpoint for an Azure AI services resource in your virtual network, Azure sends a consent request for approval to the Azure AI services resource owner. If the user who requests the creation of the private endpoint is also an owner of the resource, this consent request is automatically approved.

Azure AI services resource owners can manage consent requests and the private endpoints through the **Private endpoint connection** tab for the Azure AI services resource in the Azure portal.

Specify private endpoints

When you create a private endpoint, specify the Azure AI services resource that it connects to. For more information on creating a private endpoint, see:

- Create a private endpoint by using the Azure portal
- Create a private endpoint by using Azure PowerShell
- Create a private endpoint by using the Azure CLI

Connect to private endpoints

> **Note**
>
> Azure OpenAI Service uses a different private DNS zone and public DNS zone forwarder than other Azure AI services. For the correct zone and forwarder names, see Azure services DNS zone configuration.

Clients on a virtual network that use the private endpoint use the same connection string for the Azure AI services resource as clients connecting to the public endpoint. The exception is the Speech service, which requires a separate endpoint. For more information, see Use private endpoints with the Speech service in this article. DNS resolution automatically routes the connections from the virtual network to the Azure AI services resource over a private link.

By default, Azure creates a private DNS zone attached to the virtual network with the necessary updates for the private endpoints. If you use your own DNS server, you might need to make more changes to your DNS configuration. For updates that might be required for private endpoints, see Apply DNS changes for private endpoints in this article.

Use private endpoints with the Speech service

See Use Speech service through a private endpoint.

Apply DNS changes for private endpoints

When you create a private endpoint, the DNS `CNAME` resource record for the Azure AI services resource is updated to an alias in a subdomain with the prefix `privatelink`. By default, Azure also creates a private DNS zone that corresponds to the `privatelink` subdomain, with the DNS A resource records for the private endpoints. For more information, see What is Azure Private DNS.

When you resolve the endpoint URL from outside the virtual network with the private endpoint, it resolves to the public endpoint of the Azure AI services resource. When it's resolved from the virtual network hosting the private endpoint, the endpoint URL resolves to the private endpoint's IP address.

This approach enables access to the Azure AI services resource using the same connection string for clients in the virtual network that hosts the private endpoints and clients outside the virtual network.

If you use a custom DNS server on your network, clients must be able to resolve the fully qualified domain name (FQDN) for the Azure AI services resource endpoint to the private endpoint IP address. Configure your DNS server to delegate your private link subdomain to the private DNS zone for the virtual network.

> **Tip**
>
> When you use a custom or on-premises DNS server, you should configure your DNS server to resolve the Azure AI services resource name in the `privatelink` subdomain to the private endpoint IP address. Delegate the `privatelink` subdomain to the private DNS zone of the virtual network. Alternatively, configure the DNS zone on your DNS server and add the DNS A records.

For more information on configuring your own DNS server to support private endpoints, see the following resources:

- Name resolution that uses your own DNS server
- DNS configuration

Grant access to trusted Azure services for Azure OpenAI

You can grant a subset of trusted Azure services access to Azure OpenAI, while maintaining network rules for other apps. These trusted services will then use managed identity to authenticate your Azure OpenAI service. The following table lists the services that can access Azure OpenAI if the managed identity of those services have the appropriate role assignment.

Expand table

Service	Resource provider name
Azure AI Services	`Microsoft.CognitiveServices`
Azure Machine Learning	`Microsoft.MachineLearningServices`
Azure AI Search	`Microsoft.Search`

You can grant networking access to trusted Azure services by creating a network rule exception using the REST API:

```
Bash
```

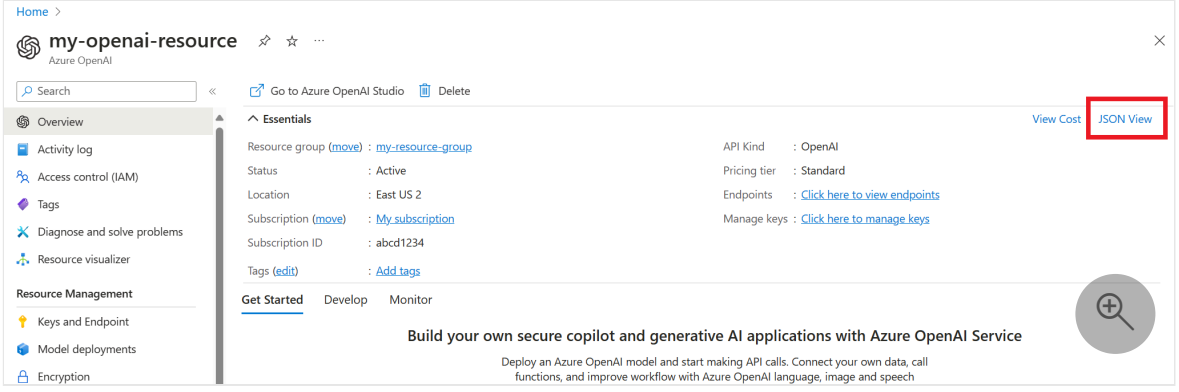
```
accessToken=$(az account get-access-token --resource
https://management.azure.com --query "accessToken" --output tsv)
rid="/subscriptions/<your subscription id>/resourceGroups/<your resource
group>/providers/Microsoft.CognitiveServices/accounts/<your Azure AI
resource name>"

curl -i -X PATCH https://management.azure.com$rid?api-version=2023-10-01-
preview \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $accessToken" \
-d \
'
{
    "properties":
    {
        "networkAcls": {
            "bypass": "AzureServices"
        }
    }
}
'
```

> **Note**
>
> The trusted service feature is only available using the command line described above, and cannot be done using the Azure portal.

To revoke the exception, set `networkAcls.bypass` to `None`.

To verify if the trusted service has been enabled from the Azure portal,

1. Use the **JSON View** from the Azure OpenAI resource overview page



2. Choose your latest API version under **API versions**. Only the latest API version is supported, `2023-10-01-preview` .

Pricing

For pricing details, see Azure Private Link pricing .

Next steps

- Explore the various Azure AI services
- Learn more about Virtual Network service endpoints

Content Moderator encryption of data at rest

Article • 01/18/2024

Content Moderator automatically encrypts your data when it is persisted to the cloud, helping to meet your organizational security and compliance goals.

About Azure AI services encryption

Data is encrypted and decrypted using FIPS 140-2 -compliant 256-bit AES encryption. Encryption and decryption are transparent, meaning encryption and access are managed for you. Your data is secure by default. You don't need to modify your code or applications to take advantage of encryption.

About encryption key management

By default, your subscription uses Microsoft-managed encryption keys. You can also manage your subscription with your own keys, which are called customer-managed keys. When you use customer-managed keys, you have greater flexibility in the way you create, rotate, disable, and revoke access controls. You can also audit the encryption keys that you use to protect your data. If customer-managed keys are configured for your subscription, double encryption is provided. With this second layer of protection, you can control the encryption key through your Azure Key Vault.

> **Important**
>
> Customer-managed keys are only available on the E0 pricing tier. To request the ability to use customer-managed keys, fill out and submit the **Content Moderator Customer-Managed Key Request Form** . It will take approximately 3-5 business days to hear back on the status of your request. Depending on demand, you may be placed in a queue and approved as space becomes available. Once approved for using CMK with the Content Moderator service, you will need to create a new Content Moderator resource and select E0 as the Pricing Tier. Once your Content Moderator resource with the E0 pricing tier is created, you can use Azure Key Vault to set up your managed identity.

Customer-managed keys are available in all Azure regions.

Customer-managed keys with Azure Key Vault

When you use customer-managed keys, you must use Azure Key Vault to store them. You can either create your own keys and store them in a key vault, or you can use the Key Vault APIs to generate keys. The Azure AI services resource and the key vault must be in the same region and in the same Microsoft Entra tenant, but they can be in different subscriptions. For more information about Key Vault, see What is Azure Key Vault?.

When you create a new Azure AI services resource, it's always encrypted by using Microsoft-managed keys. It's not possible to enable customer-managed keys when you create the resource. Customer-managed keys are stored in Key Vault. The key vault needs to be provisioned with access policies that grant key permissions to the managed identity that's associated with the Azure AI services resource. The managed identity is available only after the resource is created by using the pricing tier that's required for customer-managed keys.

Enabling customer-managed keys also enables a system-assigned managed identity, a feature of Microsoft Entra ID. After the system-assigned managed identity is enabled, this resource is registered with Microsoft Entra ID. After being registered, the managed identity is given access to the key vault that's selected during customer-managed key setup.

> **ⓘ Important**
>
> If you disable system-assigned managed identities, access to the key vault is removed and any data that's encrypted with the customer keys is no longer accessible. Any features that depend on this data stop working.

> **ⓘ Important**
>
> Managed identities don't currently support cross-directory scenarios. When you configure customer-managed keys in the Azure portal, a managed identity is automatically assigned behind the scenes. If you subsequently move the subscription, resource group, or resource from one Microsoft Entra directory to another, the managed identity that's associated with the resource isn't transferred to the new tenant, so customer-managed keys might no longer work. For more information, see **Transferring a subscription between Microsoft Entra directories** in **FAQs and known issues with managed identities for Azure resources**.

Configure Key Vault

When you use customer-managed keys, you need to set two properties in the key vault, **Soft Delete** and **Do Not Purge**. These properties aren't enabled by default, but you can enable them on a new or existing key vault by using the Azure portal, PowerShell, or Azure CLI.

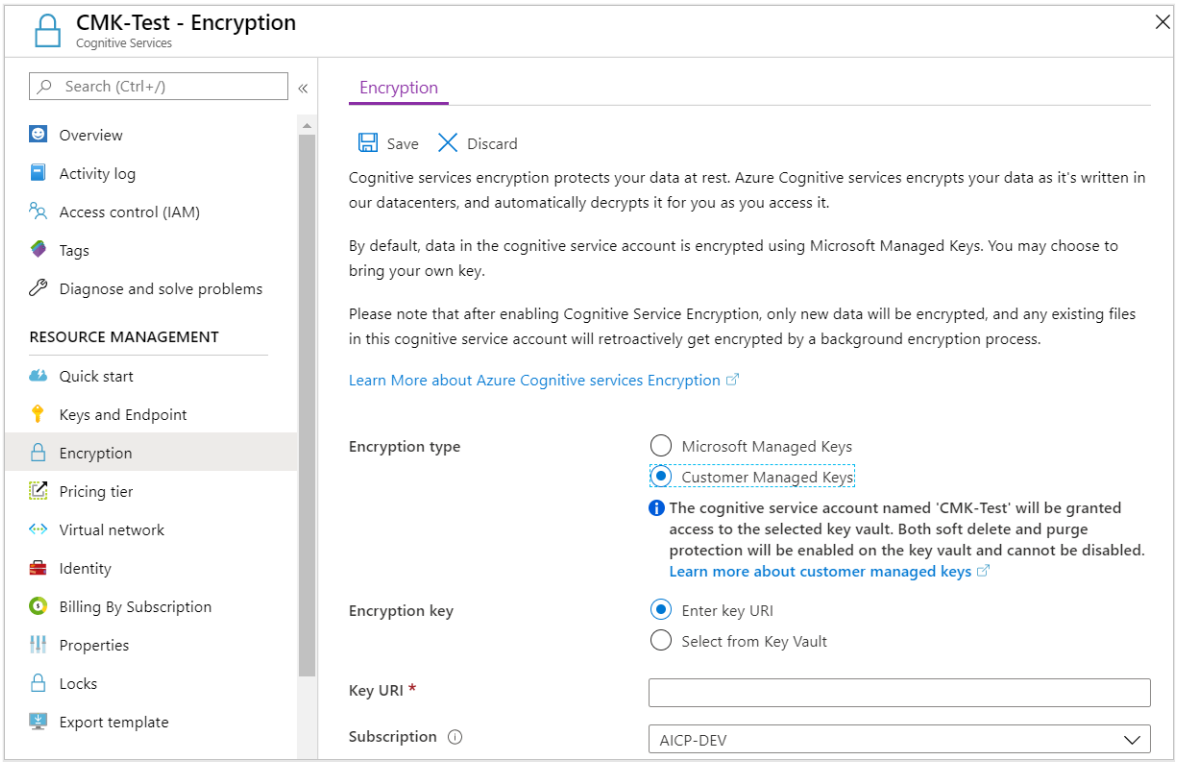
> **ⓘ Important**
>
> If the **Soft Delete** and **Do Not Purge** properties aren't enabled and you delete your key, you can't recover the data in your Azure AI services resource.

To learn how to enable these properties on an existing key vault, see Azure Key Vault recovery management with soft delete and purge protection.

Enable customer-managed keys for your resource

To enable customer-managed keys in the Azure portal, follow these steps:

1. Go to your Azure AI services resource.
2. On the left, select **Encryption**.
3. Under **Encryption type**, select **Customer Managed Keys**, as shown in the following screenshot.



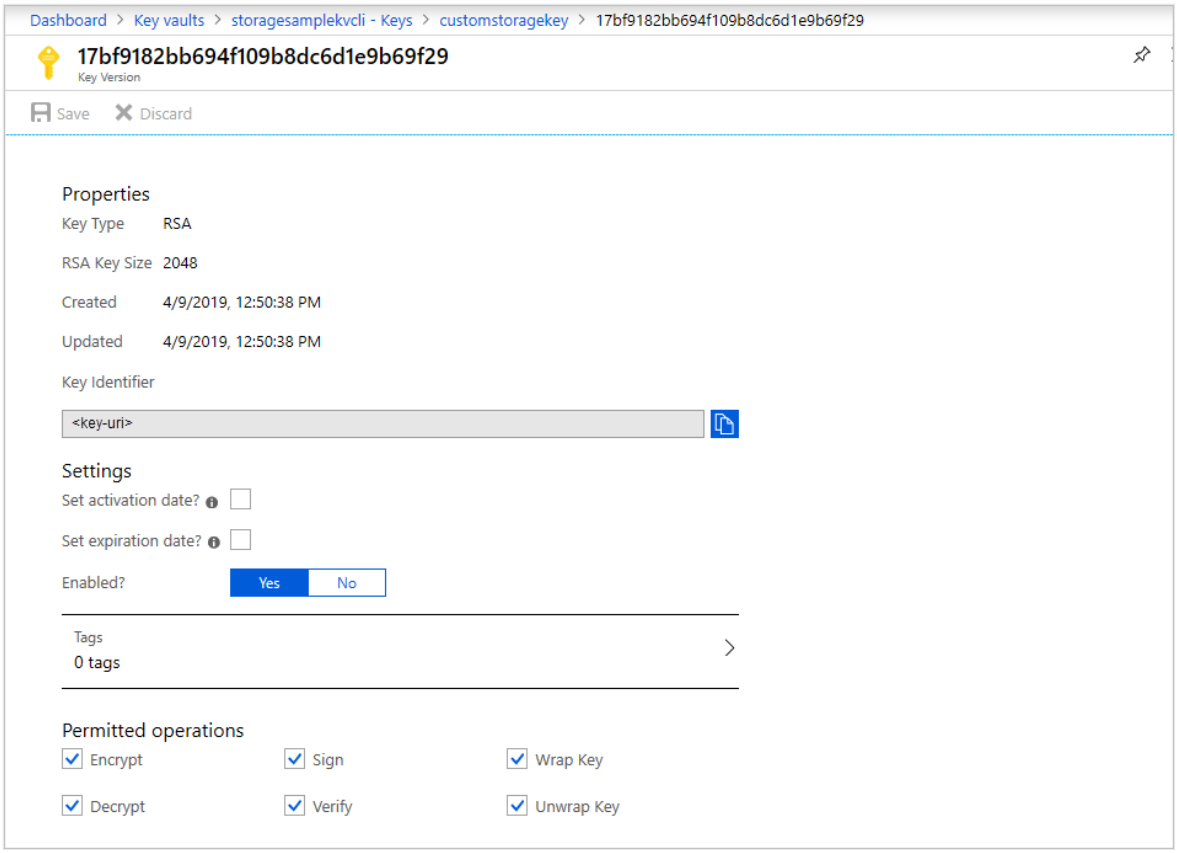
Specify a key

After you enable customer-managed keys, you can specify a key to associate with the Azure AI services resource.

Specify a key as a URI

To specify a key as a URI, follow these steps:

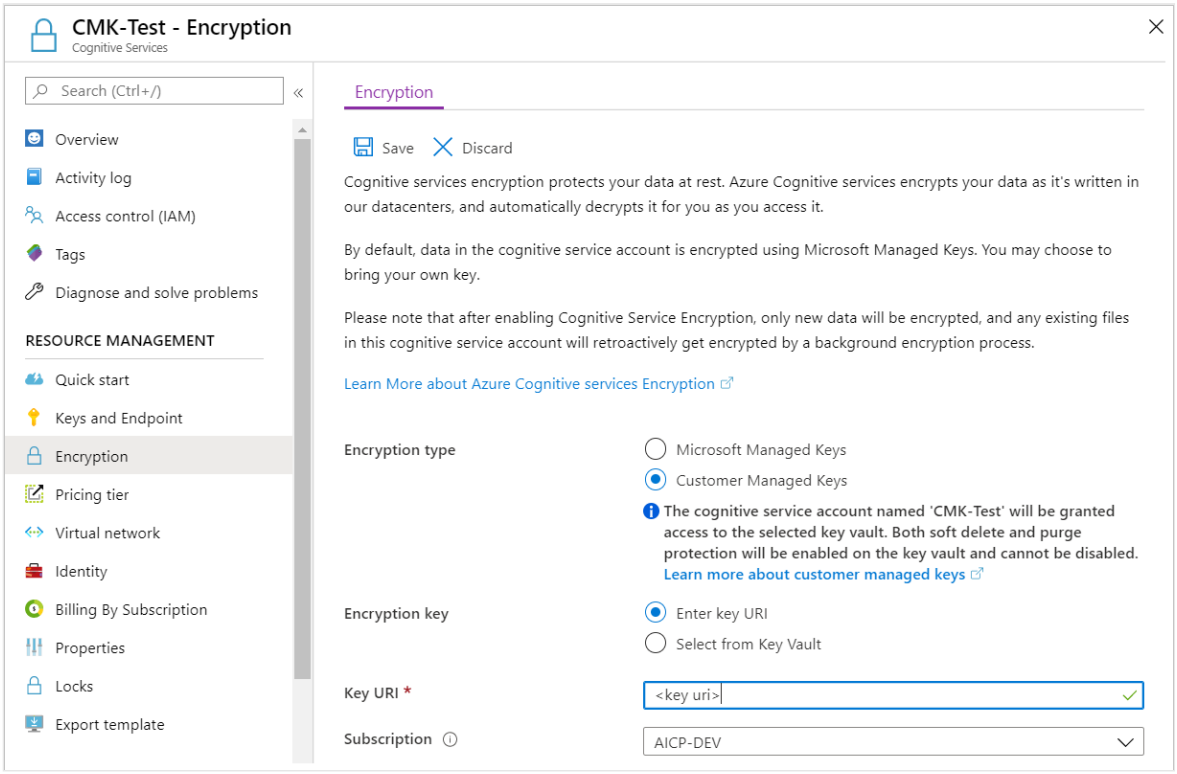
1. In the Azure portal, go to your key vault.
2. Under **Settings**, select **Keys**.
3. Select the desired key, and then select the key to view its versions. Select a key version to view the settings for that version.
4. Copy the **Key Identifier** value, which provides the URI.



5. Go back to your Azure AI services resource, and then select **Encryption**.

6. Under **Encryption key**, select **Enter key URI**.

7. Paste the URI that you copied into the **Key URI** box.



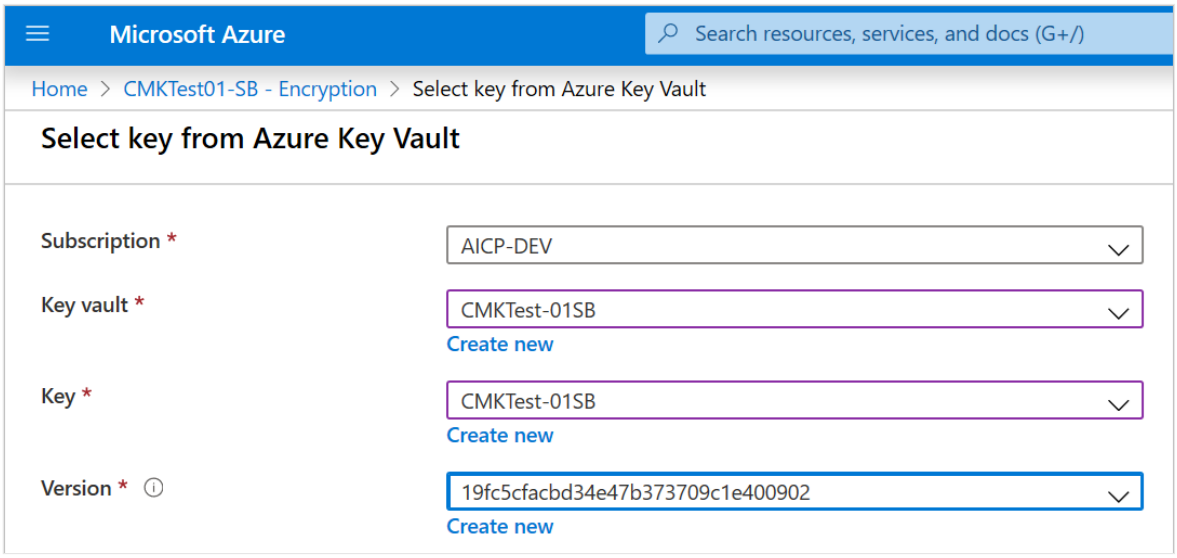
8. Under **Subscription**, select the subscription that contains the key vault.

9. Save your changes.

Specify a key from a key vault

To specify a key from a key vault, first make sure that you have a key vault that contains a key. Then follow these steps:

1. Go to your Azure AI services resource, and then select **Encryption**.
2. Under **Encryption key**, select **Select from Key Vault**.
3. Select the key vault that contains the key that you want to use.
4. Select the key that you want to use.



5. Save your changes.

Update the key version

When you create a new version of a key, update the Azure AI services resource to use the new version. Follow these steps:

1. Go to your Azure AI services resource, and then select **Encryption**.
2. Enter the URI for the new key version. Alternately, you can select the key vault and then select the key again to update the version.
3. Save your changes.

Use a different key

To change the key that you use for encryption, follow these steps:

1. Go to your Azure AI services resource, and then select **Encryption**.

2. Enter the URI for the new key. Alternately, you can select the key vault and then select a new key.
3. Save your changes.

Rotate customer-managed keys

You can rotate a customer-managed key in Key Vault according to your compliance policies. When the key is rotated, you must update the Azure AI services resource to use the new key URI. To learn how to update the resource to use a new version of the key in the Azure portal, see Update the key version.

Rotating the key doesn't trigger re-encryption of data in the resource. No further action is required from the user.

Revoke access to customer-managed keys

To revoke access to customer-managed keys, use PowerShell or Azure CLI. For more information, see Azure Key Vault PowerShell or Azure Key Vault CLI. Revoking access effectively blocks access to all data in the Azure AI services resource, because the encryption key is inaccessible by Azure AI services.

Disable customer-managed keys

When you disable customer-managed keys, your Azure AI services resource is then encrypted with Microsoft-managed keys. To disable customer-managed keys, follow these steps:

1. Go to your Azure AI services resource, and then select **Encryption**.
2. Clear the checkbox that's next to **Use your own key**.

Next steps

- For a full list of services that support CMK, see Customer-Managed Keys for Azure AI services
- What is Azure Key Vault?
- Azure AI services Customer-Managed Key Request Form

Learn image moderation concepts

Article • 01/18/2024

Use Content Moderator's machine-assisted image moderation to moderate images for adult and racy content. Scan images for text content and extract that text, and detect faces. You can match images against custom lists, and take further action.

Evaluating for adult and racy content

The **Evaluate** operation returns a confidence score between 0 and 1. It also returns boolean data equal to true or false. These values predict whether the image contains potential adult or racy content. When you call the API with your image (file or URL), the returned response includes the following information:

JSON

```
"ImageModeration": {
    ............
    "adultClassificationScore": 0.019196987152099609,
    "isImageAdultClassified": false,
    "racyClassificationScore": 0.032390203326940536,
    "isImageRacyClassified": false,
    ............
    ],
```

> **Note**
>
> - `isImageAdultClassified` represents the potential presence of images that may be considered sexually explicit or adult in certain situations.
> - `isImageRacyClassified` represents the potential presence of images that may be considered sexually suggestive or mature in certain situations.
> - The scores are between 0 and 1. The higher the score, the higher the model is predicting that the category may be applicable. This preview relies on a statistical model rather than manually coded outcomes. We recommend testing with your own content to determine how each category aligns to your requirements.
> - The boolean values are either true or false depending on the internal score thresholds. Customers should assess whether to use this value or decide on custom thresholds based on their content policies.

Detecting text with Optical Character Recognition (OCR)

The **Optical Character Recognition (OCR)** operation predicts the presence of text content in an image and extracts it for text moderation, among other uses. You can specify the language. If you do not specify a language, the detection defaults to English.

The response includes the following information:

- The original text.
- The detected text elements with their confidence scores.

Example extract:

JSON


```
"TextDetection": {
    "status": {
        "code": 3000.0,
        "description": "OK",
        "exception": null
    },
    .........
    "language": "eng",
    "text": "IF WE DID \r\nALL \r\nTHE THINGS \r\nWE ARE \r\nCAPABLE \r\nOF
DOING, \r\nWE WOULD \r\nLITERALLY \r\nASTOUND \r\nOURSELVE \r\n",
    "candidates": []
},
```


Detecting faces

Detecting faces helps to detect personal data such as faces in the images. You detect potential faces and the number of potential faces in each image.

A response includes this information:

- Faces count
- List of locations of faces detected

Example extract:

JSON


```
"FaceDetection": {
    ......
    "result": true,
```

```
    "count": 2,
    "advancedInfo": [
        .....
    ],
    "faces": [
        {
            "bottom": 598,
            "left": 44,
            "right": 268,
            "top": 374
        },
        {
            "bottom": 620,
            "left": 308,
            "right": 532,
            "top": 396
        }
    ]
}
```

Creating and managing custom lists

In many online communities, after users upload images or other type of content, offensive items may get shared multiple times over the following days, weeks, and months. The costs of repeatedly scanning and filtering out the same image or even slightly modified versions of the image from multiple places can be expensive and error-prone.

Instead of moderating the same image multiple times, you add the offensive images to your custom list of blocked content. That way, your content moderation system compares incoming images against your custom lists and stops any further processing.

> ⓘ **Note**
>
> There is a maximum limit of **5 image lists** with each list to **not exceed 10,000 images**.

The Content Moderator provides a complete Image List Management API with operations for managing lists of custom images. Start with the Image Lists API Console and use the REST API code samples. Also check out the Image List .NET quickstart if you are familiar with Visual Studio and C#.

Matching against your custom lists

The Match operation allows fuzzy matching of incoming images against any of your custom lists, created and managed using the List operations.

If a match is found, the operation returns the identifier and the moderation tags of the matched image. The response includes this information:

- Match score (between 0 and 1)
- Matched image
- Image tags (assigned during previous moderation)
- Image labels

Example extract:

JSON


```
{
    ..............,
    "IsMatch": true,
    "Matches": [
        {
            "Score": 1.0,
            "MatchId": 169490,
            "Source": "169642",
            "Tags": [],
            "Label": "Sports"
        }
    ],
    ....
}
```


Next steps

Test drive the Image Moderation API console and use the REST API code samples.

Learn text moderation concepts

Article • 05/13/2024

Use Content Moderator's text moderation models to analyze text content, such as chat rooms, discussion boards, chatbots, e-commerce catalogs, and documents.

The service response includes the following information:

- Profanity: term-based matching with built-in list of profane terms in various languages
- Classification: machine-assisted classification into three categories
- Personal data
- Auto-corrected text
- Original text
- Language

Profanity

If the API detects any profane terms in any of the supported languages, those terms are included in the response. The response also contains their location (`Index`) in the original text. The `ListId` in the following sample JSON refers to terms found in custom term lists if available.

JSON

```
"Terms": [
    {
        "Index": 118,
        "OriginalIndex": 118,
        "ListId": 0,
        "Term": "<offensive word>"
    }
```

> **Note**
>
> For the **language** parameter, assign `eng` or leave it empty to see the machine-assisted **classification** response (preview feature). **This feature supports English only**.
>
> For **profanity terms** detection, use the ISO 639-3 code of the supported languages listed in this article, or leave it empty.

Classification

Content Moderator's machine-assisted **text classification feature** supports **English only**, and helps detect potentially undesired content. The flagged content may be assessed as inappropriate depending on context. It conveys the likelihood of each category. The feature uses a trained model to identify possible abusive, derogatory or discriminatory language. This includes slang, abbreviated words, offensive, and intentionally misspelled words.

The following extract in the JSON extract shows an example output:

JSON

```
"Classification": {
    "ReviewRecommended": true,
    "Category1": {
        "Score": 1.5113095059859916E-06
    },
    "Category2": {
        "Score": 0.12747249007225037
    },
    "Category3": {
        "Score": 0.98799997568130493
    }
}
```

Explanation

- `Category1` refers to potential presence of language that may be considered sexually explicit or adult in certain situations.
- `Category2` refers to potential presence of language that may be considered sexually suggestive or mature in certain situations.
- `Category3` refers to potential presence of language that may be considered offensive in certain situations.
- `Score` is between 0 and 1. The higher the score, the higher the model is predicting that the category may be applicable. This feature relies on a statistical model rather than manually coded outcomes. We recommend testing with your own content to determine how each category aligns to your requirements.
- `ReviewRecommended` is either true or false depending on the internal score thresholds. Customers should assess whether to use this value or decide on custom thresholds based on their content policies.

Personal data

The personal data feature detects the potential presence of this information:

- Email address
- US mailing address
- IP address
- US phone number

The following example shows a sample response:

JSON


```
"pii":{
  "email":[
      {
        "detected":"abcdef@abcd.com",
        "sub_type":"Regular",
        "text":"abcdef@abcd.com",
        "index":32
      }
  ],
  "ssn":[

  ],
  "ipa":[
      {
        "sub_type":"IPV4",
        "text":"255.255.255.255",
        "index":72
      }
  ],
  "phone":[
      {
        "country_code":"US",
        "text":"6657789887",
        "index":56
      }
  ],
  "address":[
      {
        "text":"1 Microsoft Way, Redmond, WA 98052",
        "index":89
      }
  ]
}
```


Auto-correction

The text moderation response can optionally return the text with basic auto-correction applied.

For example, the following input text has a misspelling.

> The quick brown fox jumps over the lazzy dog.

If you specify auto-correction, the response contains the corrected version of the text:

> The quick brown fox jumps over the lazy dog.

Creating and managing your custom lists of terms

While the default, global list of terms works great for most cases, you may want to screen against terms that are specific to your business needs. For example, you may want to filter out any competitive brand names from posts by users.

> **Note**
>
> There is a maximum limit of **5 term lists** with each list to **not exceed 10,000 terms**.

The following example shows the matching List ID:

JSON

```
"Terms": [
    {
        "Index": 118,
        "OriginalIndex": 118,
        "ListId": 231.
        "Term": "<offensive word>"
    }
```

The Content Moderator provides a Term List API with operations for managing custom term lists. Check out the Term Lists .NET quickstart if you are familiar with Visual Studio and C#.

Next steps

Test out the APIs with the Quickstart.

Content Moderator API reference

Article • 01/18/2024

You can get started with Azure Content Moderator APIs by doing the following:

- In the Azure portal, subscribe to the Content Moderator API.

You can use the following **Content Moderator APIs** to set up your post-moderation workflows.

Expand table

Description	Reference
Image Moderation API Scan images and detect potential adult and racy content by using tags, confidence scores, and other extracted information.	Image Moderation API reference
Text Moderation API Scan text content. Profanity terms and personal data are returned.	Text Moderation API reference
Video Moderation API Scan videos and detect potential adult and racy content.	Video Moderation API overview
List Management API Create and manage custom exclusion or inclusion lists of images and text. If enabled, the **Image - Match** and **Text - Screen** operations do fuzzy matching of the submitted content against your custom lists. For efficiency, you can skip the machine learning-based moderation step.	List Management API reference

Azure Content Moderator SDK for .NET - latest

Article • 02/14/2023

Packages - latest

Reference	Package	Source
Content Moderator	Microsoft.Azure.CognitiveServices.ContentModerator	GitHub

Content Moderator

Reference

Packages

Expand table

com.microsoft.azure.cognitiveservices.vision.contentmoderator
com.microsoft.azure.cognitiveservices.vision.contentmoderator.models

@azure/cognitiveservices-contentmoderator package

Reference

Classes

Expand table

ContentModeratorClient	
ContentModeratorClientContext	
ImageModeration	Class representing a ImageModeration.
ListManagementImage	Class representing a ListManagementImage.
ListManagementImageLists	Class representing a ListManagementImageLists.
ListManagementTerm	Class representing a ListManagementTerm.
ListManagementTermLists	Class representing a ListManagementTermLists.
Reviews	Class representing a Reviews.
TextModeration	Class representing a TextModeration.

Interfaces

Expand table

Address	Address details.
APIError	Error information returned by the API
Body	An interface representing Body.
Candidate	OCR candidate text.
Classification	The classification details of the text.
Classification Category1	The category1 score details of the text. Click here for more details on category classification.
Classification Category2	The category2 score details of the text. Click here for more details on category classification.

Classification Category3	The category3 score details of the text. Click here for more details on category classification.
Content	An interface representing Content.
CreateReviewBody Item	Schema items of the body.
CreateReviewBody ItemMetadataItem	An interface representing CreateReviewBodyItemMetadataItem.
CreateVideoReviews BodyItem	Schema items of the body.
CreateVideoReviews BodyItemMetadata Item	An interface representing CreateVideoReviewsBodyItemMetadataItem.
CreateVideoReviews BodyItemVideo FramesItem	An interface representing CreateVideoReviewsBodyItemVideoFramesItem.
CreateVideoReviews BodyItemVideo FramesItem MetadataItem	An interface representing CreateVideoReviewsBodyItemVideoFramesItemMetadataItem.
CreateVideoReviews BodyItemVideo FramesItemReviewer ResultTagsItem	An interface representing CreateVideoReviewsBodyItemVideoFramesItemReviewerResultTagsItem.
DetectedLanguage	Detect language result.
DetectedTerms	Detected Terms details.
Email	Email Address details.
ErrorModel	Error body.
Evaluate	Evaluate response object.
Face	Coordinates to the found face.
FoundFaces	Request object the contains found faces.
Frame	Video frame property details.
Frames	The response for a Get Frames request.
Image	Image Properties.

ImageAdditionalInfoItem	An interface representing ImageAdditionalInfoItem.
ImageIds	Image Id properties.
ImageList	Image List Properties.
ImageModerationEvaluateFileInputOptionalParams	Optional Parameters.
ImageModerationEvaluateMethodOptionalParams	Optional Parameters.
ImageModerationEvaluateUrlInputOptionalParams	Optional Parameters.
ImageModerationFindFacesFileInputOptionalParams	Optional Parameters.
ImageModerationFindFacesOptionalParams	Optional Parameters.
ImageModerationFindFacesUrlInputOptionalParams	Optional Parameters.
ImageModerationMatchFileInputOptionalParams	Optional Parameters.
ImageModerationMatchMethodOptionalParams	Optional Parameters.
ImageModerationMatchUrlInputOptionalParams	Optional Parameters.
ImageModerationOCRFileInputOptionalParams	Optional Parameters.
ImageModerationOCRMethodOptionalParams	Optional Parameters.

ImageModerationOCRUrlInputOptionalParams	Optional Parameters.
ImageUrl	An interface representing ImageUrl.
IPA	IP Address details.
Job	The Job object.
JobExecutionReportDetails	Job Execution Report Values.
JobId	An interface representing JobId.
JobListResult	The list of job ids.
KeyValuePair	The key value pair object properties.
ListManagementImageAddImageFileInputOptionalParams	Optional Parameters.
ListManagementImageAddImageOptionalParams	Optional Parameters.
ListManagementImageAddImageUrlInputOptionalParams	Optional Parameters.
ListManagementTermGetAllTermsOptionalParams	Optional Parameters.
Match	The match details.
MatchResponse	The response for a Match request.
OCR	Contains the text found in image for the language specified.
Phone	Phone Property details.
PII	Personal Identifier Information details.
RefreshIndex	Refresh Index Response.
Review	The Review object.
ReviewsAddVideoFrameOptional	Optional Parameters.

Params	
ReviewsAddVideoFrameStreamOptionalParams	Optional Parameters.
ReviewsAddVideoFrameUrlOptionalParams	Optional Parameters.
ReviewsCreateJobOptionalParams	Optional Parameters.
ReviewsCreateReviewsOptionalParams	Optional Parameters.
ReviewsCreateVideoReviewsOptionalParams	Optional Parameters.
ReviewsGetVideoFramesOptionalParams	Optional Parameters.
Screen	The response for a Screen text request.
SSN	Detected SSN details.
Status	Status properties.
Tag	Tag details.
TermList	Term List Properties.
Terms	Terms properties.
TermsData	All term Id response properties.
TermsInList	Terms in list Id passed.
TermsPaging	Paging details.
TextModerationScreenTextOptionalParams	Optional Parameters.
TranscriptModerationBodyItem	Schema items of the body.
TranscriptModerationBody	An interface representing TranscriptModerationBodyItemTermsItem.

ItemTermsItem	
VideoFrameBodyItem	Schema items of the body.
VideoFrameBodyItemMetadataItem	An interface representing VideoFrameBodyItemMetadataItem.
VideoFrameBodyItemReviewerResultTagsItem	An interface representing VideoFrameBodyItemReviewerResultTagsItem.

Type Aliases

Expand table

ContentType	Defines values for ContentType. Possible values include: 'Image', 'Text', 'Video'
ImageModerationEvaluateFileInputResponse	Contains response data for the evaluateFileInput operation.
ImageModerationEvaluateMethodResponse	Contains response data for the evaluateMethod operation.
ImageModerationEvaluateUrlInputResponse	Contains response data for the evaluateUrlInput operation.
ImageModerationFindFacesFileInputResponse	Contains response data for the findFacesFileInput operation.
ImageModerationFindFacesResponse	Contains response data for the findFaces operation.
ImageModerationFindFacesUrlInputResponse	Contains response data for the findFacesUrlInput operation.
ImageModerationMatchFileInputResponse	Contains response data for the matchFileInput operation.
ImageModerationMatchMethodResponse	Contains response data for the matchMethod operation.
ImageModerationMatchUrlInputResponse	Contains response data for the matchUrlInput operation.
ImageModerationOCRFileInputResponse	Contains response data for the oCRFileInput operation.

ImageModerationOCRMethodResponse	Contains response data for the oCRMethod operation.
ImageModerationOCRUrlInputResponse	Contains response data for the oCRUrlInput operation.
JobContentType	Defines values for JobContentType. Possible values include: 'application/json', 'image/jpeg'
ListManagementImageAddImageFileInputResponse	Contains response data for the addImageFileInput operation.
ListManagementImageAddImageResponse	Contains response data for the addImage operation.
ListManagementImageAddImageUrlInputResponse	Contains response data for the addImageUrlInput operation.
ListManagementImageDeleteAllImagesResponse	Contains response data for the deleteAllImages operation.
ListManagementImageDeleteImageResponse	Contains response data for the deleteImage operation.
ListManagementImageGetAllImageIdsResponse	Contains response data for the getAllImageIds operation.
ListManagementImageListsCreateResponse	Contains response data for the create operation.
ListManagementImageListsDeleteMethodResponse	Contains response data for the deleteMethod operation.
ListManagementImageListsGetAllImageListsResponse	Contains response data for the getAllImageLists operation.
ListManagementImageListsGetDetailsResponse	Contains response data for the getDetails operation.
ListManagementImageListsRefreshIndexMethodResponse	Contains response data for the refreshIndexMethod operation.
ListManagementImageListsUpdateResponse	Contains response data for the update operation.
ListManagementTermDeleteAllTermsResponse	Contains response data for the deleteAllTerms operation.
ListManagementTermDeleteTermResponse	Contains response data for the deleteTerm operation.

ListManagementTermGetAllTermsResponse	Contains response data for the getAllTerms operation.
ListManagementTermListsCreateResponse	Contains response data for the create operation.
ListManagementTermListsDeleteMethodResponse	Contains response data for the deleteMethod operation.
ListManagementTermListsGetAllTermListsResponse	Contains response data for the getAllTermLists operation.
ListManagementTermListsGetDetailsResponse	Contains response data for the getDetails operation.
ListManagementTermListsRefreshIndexMethodResponse	Contains response data for the refreshIndexMethod operation.
ListManagementTermListsUpdateResponse	Contains response data for the update operation.
ReviewsCreateJobResponse	Contains response data for the createJob operation.
ReviewsCreateReviewsResponse	Contains response data for the createReviews operation.
ReviewsCreateVideoReviewsResponse	Contains response data for the createVideoReviews operation.
ReviewsGetJobDetailsResponse	Contains response data for the getJobDetails operation.
ReviewsGetReviewResponse	Contains response data for the getReview operation.
ReviewsGetVideoFramesResponse	Contains response data for the getVideoFrames operation.
StatusEnum	Defines values for StatusEnum. Possible values include: 'Complete', 'Unpublished', 'Pending'
TextContentType	Defines values for TextContentType. Possible values include: 'text/plain', 'text/html', 'text/xml', 'text/markdown'
TextContentType1	Defines values for TextContentType1. Possible values include: 'text/plain', 'text/html', 'text/xml', 'text/markdown'
TextModerationDetectLanguageResponse	Contains response data for the detectLanguage operation.
TextModerationScreenTextResponse	Contains response data for the screenText operation.
Type	Defines values for Type. Possible values include: 'Image', 'Text'

Azure Content Moderator SDK for Python - latest

Article • 02/10/2023

Packages - latest

Reference	Package	Source
Content Moderator	azure-cognitiveservices-vision-contentmoderator	GitHub

Azure AI services support and help options

Article • 02/22/2024

Are you just starting to explore the functionality of Azure AI services? Perhaps you are implementing a new feature in your application. Or after using the service, do you have suggestions on how to improve it? Here are options for where you can get support, stay up-to-date, give feedback, and report bugs for Azure AI services.

Create an Azure support request

Explore the range of Azure support options and choose the plan that best fits, whether you're a developer just starting your cloud journey or a large organization deploying business-critical, strategic applications. Azure customers can create and manage support requests in the Azure portal.

- Azure portal
- Azure portal for the United States government

Post a question on Microsoft Q&A

For quick and reliable answers on your technical product questions from Microsoft Engineers, Azure Most Valuable Professionals (MVPs), or our expert community, engage with us on Microsoft Q&A, Azure's preferred destination for community support.

If you can't find an answer to your problem using search, submit a new question to Microsoft Q&A. Use one of the following tags when you ask your question:

- Azure AI services

Vision

- Azure AI Vision
- Custom Vision
- Face
- Document Intelligence
- Video Indexer

Language

- Immersive Reader
- Language Understanding (LUIS)
- QnA Maker
- Language service
- Translator

Speech

- Speech service

Decision

- Anomaly Detector
- Content Moderator
- Metrics Advisor
- Personalizer

Azure OpenAI

- Azure OpenAI

Post a question to Stack Overflow

For answers on your developer questions from the largest community developer ecosystem, ask your question on Stack Overflow.

If you do submit a new question to Stack Overflow, please use one or more of the following tags when you create the question:

- Azure AI services

Vision

- Azure AI Vision
- Custom Vision
- Face
- Document Intelligence
- Video Indexer

Language

- Immersive Reader
- Language Understanding (LUIS)

- QnA Maker
- Language service
- Translator

Speech

- Speech service

Decision

- Anomaly Detector
- Content Moderator
- Metrics Advisor
- Personalizer

Azure OpenAI

- Azure OpenAI

Submit feedback

To request new features, post them on https://feedback.azure.com. Share your ideas for making Azure AI services and its APIs work better for the applications you develop.

- Azure AI services

Vision

- Azure AI Vision
- Custom Vision
- Face
- Document Intelligence
- Video Indexer

Language

- Immersive Reader
- Language Understanding (LUIS)
- QnA Maker
- Language service
- Translator

Speech

- Speech service

Decision

- Anomaly Detector
- Content Moderator
- Metrics Advisor
- Personalizer

Stay informed

Staying informed about features in a new release or news on the Azure blog can help you find the difference between a programming error, a service bug, or a feature not yet available in Azure AI services.

- Learn more about product updates, roadmap, and announcements in Azure Updates.
- News about Azure AI services is shared in the Azure blog.
- Join the conversation on Reddit about Azure AI services.

Next steps

What are Azure AI services?

Compare Azure Government and global Azure

Article • 10/12/2023

Microsoft Azure Government uses same underlying technologies as global Azure, which includes the core components of Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Both Azure and Azure Government have the same comprehensive security controls in place and the same Microsoft commitment on the safeguarding of customer data. Whereas both cloud environments are assessed and authorized at the FedRAMP High impact level, Azure Government provides an extra layer of protection to customers through contractual commitments regarding storage of customer data in the United States and limiting potential access to systems processing customer data to screened US persons. These commitments may be of interest to customers using the cloud to store or process data subject to US export control regulations.

> **Note**
>
> These lists and tables do not include feature or bundle availability in the Azure Government Secret or Azure Government Top Secret clouds. For more information about specific availability for air-gapped clouds, please contact your account team.

Export control implications

You're responsible for designing and deploying your applications to meet US export control requirements such as the requirements prescribed in the EAR, ITAR, and DoE 10 CFR Part 810. In doing so, you shouldn't include sensitive or restricted information in Azure resource names, as explained in Considerations for naming Azure resources.

Guidance for developers

Most of the currently available technical content assumes that applications are being developed on global Azure rather than on Azure Government. For this reason, it's important to be aware of two key differences in applications that you develop for hosting in Azure Government.

- Certain services and features that are in specific regions of global Azure might not be available in Azure Government.
- Feature configurations in Azure Government might differ from those in global Azure.

Therefore, it's important to review your sample code and configurations to ensure that you are building within the Azure Government cloud services environment.

For more information, see Azure Government developer guide.

> **Note**
>
> This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see **Introducing the new Azure PowerShell Az module**. For Az module installation instructions, see **Install the Azure Az PowerShell module**.

You can use AzureCLI or PowerShell to obtain Azure Government endpoints for services you provisioned:

- Use **Azure CLI** to run the az cloud show command and provide `AzureUSGovernment` as the name of the target cloud environment. For example,

Azure CLI

```
az cloud show --name AzureUSGovernment
```

should get you different endpoints for Azure Government.

- Use a **PowerShell** cmdlet such as Get-AzEnvironment to get endpoints and metadata for an instance of Azure service. For example,

  PowerShell

  ```
  Get-AzEnvironment -Name AzureUSGovernment
  ```

  should get you properties for Azure Government. This cmdlet gets environments from your subscription data file.

Table below lists API endpoints in Azure vs. Azure Government for accessing and managing some of the more common services. If you provisioned a service that isn't listed in the table below, see the Azure CLI and PowerShell examples above for suggestions on how to obtain the corresponding Azure Government endpoint.

Expand table

Service category	Service name	Azure Public	Azure Government	Notes
AI + machine learning	Azure Bot Service	botframework.com	botframework.azure.us	
	Azure AI Document Intelligence	cognitiveservices.azure.com	cognitiveservices.azure.us	
	Azure OpenAI Service	openai.azure.com	openai.azure.us	
	Computer Vision	cognitiveservices.azure.com	cognitiveservices.azure.us	
	Custom Vision	cognitiveservices.azure.com	cognitiveservices.azure.us Portal	
	Content Moderator	cognitiveservices.azure.com	cognitiveservices.azure.us	
	Face API	cognitiveservices.azure.com	cognitiveservices.azure.us	
	Language Understanding	cognitiveservices.azure.com	cognitiveservices.azure.us Portal	Part of Azure AI Language
	Personalizer	cognitiveservices.azure.com	cognitiveservices.azure.us	
	QnA Maker	cognitiveservices.azure.com	cognitiveservices.azure.us	Part of Azure AI Language
	Speech service	See STT API docs	Speech Studio See Speech service endpoints **Speech translation endpoints**	

Service category	Service name	Azure Public	Azure Government	Notes
			Virginia: `https://usgovvirginia.s2s.speech.azure.us` Arizona: `https://usgovarizona.s2s.speech.azure.us`	
	Text Analytics	cognitiveservices.azure.com	cognitiveservices.azure.us	Part of Azure AI Language
	Translator	See Translator API docs	cognitiveservices.azure.us	
Analytics	Azure HDInsight	azurehdinsight.net	azurehdinsight.us	
	Event Hubs	servicebus.windows.net	servicebus.usgovcloudapi.net	
	Power BI	app.powerbi.com	app.powerbigov.us	Power BI US Gov
Compute	Batch	batch.azure.com	batch.usgovcloudapi.net	
	Cloud Services	cloudapp.net	usgovcloudapp.net	
Containers	Azure Service Fabric	cloudapp.azure.com	cloudapp.usgovcloudapi.net	
	Container Registry	azurecr.io	azurecr.us	
Databases	Azure Cache for Redis	redis.cache.windows.net	redis.cache.usgovcloudapi.net	See How to connect to other clouds
	Azure Cosmos DB	documents.azure.com	documents.azure.us	
	Azure Database for MariaDB	mariadb.database.azure.com	mariadb.database.usgovcloudapi.net	
	Azure Database for MySQL	mysql.database.azure.com	mysql.database.usgovcloudapi.net	
	Azure Database for PostgreSQL	postgres.database.azure.com	postgres.database.usgovcloudapi.net	
	Azure SQL Database	database.windows.net	database.usgovcloudapi.net	
Identity	Microsoft Entra ID	login.microsoftonline.com	login.microsoftonline.us	
		certauth.login.microsoftonline.com	certauth.login.microsoftonline.us	
		passwordreset.microsoftonline.com	passwordreset.microsoftonline.us	
Integration	Service Bus	servicebus.windows.net	servicebus.usgovcloudapi.net	
Internet of Things	Azure IoT Hub	azure-devices.net	azure-devices.us	

Service category	Service name	Azure Public	Azure Government	Notes
	Azure Maps	atlas.microsoft.com	atlas.azure.us	
	Notification Hubs	servicebus.windows.net	servicebus.usgovcloudapi.net	
Management and governance	Azure Automation	azure-automation.net	azure-automation.us	
	Azure Monitor	mms.microsoft.com	oms.microsoft.us	Log Analytics workspace portal
		ods.opinsights.azure.com	ods.opinsights.azure.us	Data collector API
		oms.opinsights.azure.com	oms.opinsights.azure.us	
		portal.loganalytics.io	portal.loganalytics.us	
		api.loganalytics.io	api.loganalytics.us	
		docs.loganalytics.io	docs.loganalytics.us	
		adx.monitor.azure.com	adx.monitor.azure.us	Data Explorer queries
	Azure Resource Manager	management.azure.com	management.usgovcloudapi.net	
	Gallery URL	gallery.azure.com	gallery.azure.us	
	Microsoft Azure portal	portal.azure.com	portal.azure.us	
	Microsoft Intune	enterpriseregistration.windows.net	enterpriseregistration.microsoftonline.us	Enterprise registration
		manage.microsoft.com	manage.microsoft.us	Enterprise enrollment
Migration	Azure Site Recovery	hypervrecoverymanager.windowsazure.com	hypervrecoverymanager.windowsazure.us	Site Recovery service
		backup.windowsazure.com	backup.windowsazure.us	Protection service
		blob.core.windows.net	blob.core.usgovcloudapi.net	Storing VM snapshots
Networking	Traffic Manager	trafficmanager.net	usgovtrafficmanager.net	
Security	Key Vault	vault.azure.net	vault.usgovcloudapi.net	
	Managed HSM	managedhsm.azure.net	managedhsm.usgovcloudapi.net	
Storage	Azure Backup	backup.windowsazure.com	backup.windowsazure.us	

Service category	Service name	Azure Public	Azure Government	Notes
	Blob	blob.core.windows.net	blob.core.usgovcloudapi.net	
	Queue	queue.core.windows.net	queue.core.usgovcloudapi.net	
	Table	table.core.windows.net	table.core.usgovcloudapi.net	
	File	file.core.windows.net	file.core.usgovcloudapi.net	
Virtual desktop infrastructure	Azure Virtual Desktop	See AVD docs	See AVD docs	
Web	API Management	management.azure.com	management.usgovcloudapi.net	
	API Management Gateway	azure-api.net	azure-api.us	
	API Management management	management.azure-api.net	management.azure-api.us	
	API Management Portal	portal.azure-api.net	portal.azure-api.us	
	App Configuration	azconfig.io	azconfig.azure.us	
	App Service	azurewebsites.net	azurewebsites.us	
	Azure AI Search	search.windows.net	search.windows.us	
	Azure Functions	azurewebsites.net	azurewebsites.us	

Service availability

Microsoft's goal for Azure Government is to match service availability in Azure. For service availability in Azure Government, see Products available by region. Services available in Azure Government are listed by category and whether they're Generally Available or available through Preview. If a service is available in Azure Government, that fact isn't reiterated in the rest of this article. Instead, you're encouraged to review Products available by region for the latest, up-to-date information on service availability.

In general, service availability in Azure Government implies that all corresponding service features are available to you. Variations to this approach and other applicable limitations are tracked and explained in this article based on the main service categories outlined in the online directory of Azure services. Other considerations for service deployment and usage in Azure Government are also provided.

AI + machine learning

This section outlines variations and considerations when using **Azure Bot Service**, **Azure Machine Learning**, and **Cognitive Services** in the Azure Government environment. For service availability, see Products available by region.

Azure Bot Service

The following Azure Bot Service **features aren't currently available** in Azure Government:

- Bot Framework Composer integration
- Channels (due to availability of dependent services)
  - Direct Line Speech Channel
  - Telephony Channel (Preview)
  - Microsoft Search Channel (Preview)
  - Kik Channel (deprecated)

For information on how to deploy Bot Framework and Azure Bot Service bots to Azure Government, see Configure Bot Framework bots for US Government customers.

Azure Machine Learning

For feature variations and limitations, see Azure Machine Learning feature availability across cloud regions.

Azure AI services: Content Moderator

The following Content Moderator **features aren't currently available** in Azure Government:

- Review UI and Review APIs.

Azure AI Language Understanding (LUIS)

The following Language Understanding **features aren't currently available** in Azure Government:

- Speech Requests
- Prebuilt Domains

Azure AI Language Understanding (LUIS) is part of Azure AI Language.

Azure AI Speech

For feature variations and limitations, including API endpoints, see Speech service in sovereign clouds.

Azure AI services: OpenAI Service

The following features of Azure OpenAI are available in Azure Government:

Expand table

Feature	Azure OpenAI
Models available	US Gov Arizona: GPT-4 (1106-Preview) GPT-3.5-Turbo (1106) GPT-3.5-Turbo (0125) text-embedding-ada-002 (version 2) US Gov Virginia: GPT-4 (1106-Preview) GPT-3.5-Turbo (0125) text-embedding-ada-002 (version 2)

Feature	Azure OpenAI
	Learn more in Azure OpenAI Service models
Virtual network support & private link support	Yes, unless using Azure OpenAI on your data
Managed Identity	Yes, via Microsoft Entra ID
UI experience	**Azure portal** for account & resource management **Azure OpenAI Studio** for model exploration

Next steps

- Get started by requesting access to Azure OpenAI Service in Azure Government at https://aka.ms/AOAIgovaccess
- Request quota increases for the pay-as-you-go consumption model, please fill out a separate form at https://aka.ms/AOAIGovQuota

Azure AI services: Translator

For feature variations and limitations, including API endpoints, see Translator in sovereign clouds.

Analytics

This section outlines variations and considerations when using Analytics services in the Azure Government environment. For service availability, see Products available by region.

Azure HDInsight

For secured virtual networks, you'll want to allow network security groups (NSGs) access to certain IP addresses and ports. For Azure Government, you should allow the following IP addresses (all with an Allowed port of 443):

Expand table

Region	Allowed IP addresses	Allowed port
US DoD Central	52.180.249.174 52.180.250.239	443
US DoD East	52.181.164.168 52.181.164.151	443
US Gov Texas	52.238.116.212 52.238.112.86	443
US Gov Virginia	13.72.49.126 13.72.55.55 13.72.184.124 13.72.190.110	443
US Gov Arizona	52.127.3.176 52.127.3.178	443

For a demo on how to build data-centric solutions on Azure Government using HDInsight, see Azure AI services, HDInsight, and Power BI on Azure Government.

Power BI

For usage guidance, feature variations, and limitations, see Power BI for US government customers. For a demo on how to build data-centric solutions on Azure Government using Power BI, see Azure AI services, HDInsight, and Power BI on Azure Government.

Power BI Embedded

To learn how to embed analytical content within your business process application, see Tutorial: Embed a Power BI content into your application for national clouds.

Databases

This section outlines variations and considerations when using Databases services in the Azure Government environment. For service availability, see Products available by region.

Azure Database for MySQL

The following Azure Database for MySQL **features aren't currently available** in Azure Government:

- Advanced Threat Protection

Azure Database for PostgreSQL

For Flexible Server availability in Azure Government regions, see Azure Database for PostgreSQL – Flexible Server.

The following Azure Database for PostgreSQL **features aren't currently available** in Azure Government:

- Azure Cosmos DB for PostgreSQL, formerly Azure Database for PostgreSQL – Hyperscale (Citus). For more information about supported regions, see Regional availability for Azure Cosmos DB for PostgreSQL.
- The following features of the Single Server deployment option
  - Advanced Threat Protection
  - Backup with long-term retention

Developer tools

This section outlines variations and considerations when using Developer tools in the Azure Government environment. For service availability, see Products available by region.

Enterprise Dev/Test subscription offer

- Enterprise Dev/Test subscription offer in existing or separate tenant is currently available only in Azure public as documented in Azure EA portal administration.

Identity

This section outlines variations and considerations when using Identity services in the Azure Government environment. For service availability, see Products available by region.

Microsoft Entra ID P1 and P2

For feature variations and limitations, see Cloud feature availability.

For information on how to use Power BI capabilities for collaboration between Azure and Azure Government, see Cross-cloud B2B.

The following features have known limitations in Azure Government:

- Limitations with B2B Collaboration in supported Azure US Government tenants:
  - For more information about B2B collaboration limitations in Azure Government and to find out if B2B collaboration is available in your Azure Government tenant, see Microsoft Entra B2B in government and national clouds.
- Limitations with multi-factor authentication:
  - Trusted IPs isn't supported in Azure Government. Instead, use Conditional Access policies with named locations to establish when multi-factor authentication should and shouldn't be required based off the user's current IP address.

Azure Active Directory B2C

Azure Active Directory B2C is **not available** in Azure Government.

Microsoft Authentication Library (MSAL)

The Microsoft Authentication Library (MSAL) enables developers to acquire security tokens from the Microsoft identity platform to authenticate users and access secured web APIs. For feature variations and limitations, see National clouds and MSAL.

Management and governance

This section outlines variations and considerations when using Management and Governance services in the Azure Government environment. For service availability, see Products available by region.

Automation

The following Automation **features aren't currently available** in Azure Government:

- Automation analytics solution

Azure Advisor

For feature variations and limitations, see Azure Advisor in sovereign clouds.

Azure Lighthouse

The following Azure Lighthouse **features aren't currently available** in Azure Government:

- Managed Service offers published to Azure Marketplace
- Delegation of subscriptions across a national cloud and the Azure public cloud, or across two separate national clouds, isn't supported
- Privileged Identity Management (PIM) feature isn't enabled, for example, just-in-time (JIT) / eligible authorization capability

Azure Managed Grafana

The following document contains information about Azure Managed Grafana feature availability in Azure Government: Azure Managed Grafana: Feature availability in sovereign clouds.

Azure Monitor

Azure Monitor enables the same features in both Azure and Azure Government.

- System Center Operations Manager 2019 is supported equally well in both Azure and Azure Government.

The following options are available for previous versions of System Center Operations Manager:

- Integrating System Center Operations Manager 2016 with Azure Government requires an updated Advisor management pack that is included with Update Rollup 2 or later.
- System Center Operations Manager 2012 R2 requires an updated Advisor management pack included with Update Rollup 3 or later.

For more information, see Connect Operations Manager to Azure Monitor.

Frequently asked questions

- Can I migrate data from Azure Monitor logs in Azure to Azure Government?
  - No. It isn't possible to move data or your workspace from Azure to Azure Government.
- Can I switch between Azure and Azure Government workspaces from the Operations Management Suite portal?
  - No. The portals for Azure and Azure Government are separate and don't share information.

Application Insights

Application Insights (part of Azure Monitor) enables the same features in both Azure and Azure Government. This section describes the supplemental configuration that is required to use Application Insights in Azure Government.

Visual Studio – In Azure Government, you can enable monitoring on your ASP.NET, ASP.NET Core, Java, and Node.js based applications running on Azure App Service. For more information, see Application monitoring for Azure App Service overview. In Visual Studio, go to Tools|Options|Accounts|Registered Azure Clouds|Add New Azure Cloud and select Azure US Government as the Discovery endpoint. After that, adding an account in File|Account Settings will prompt you for which cloud you want to add from.

SDK endpoint modifications – In order to send data from Application Insights to an Azure Government region, you'll need to modify the default endpoint addresses that are used by the Application Insights SDKs. Each SDK requires slightly different modifications, as described in Application Insights overriding default endpoints.

Firewall exceptions – Application Insights uses several IP addresses. You might need to know these addresses if the app that you're monitoring is hosted behind a firewall. For more information, see IP addresses used by Azure Monitor from where you can download Azure Government IP addresses.

> **Note**
>
> Although these addresses are static, it's possible that we'll need to change them from time to time. All Application Insights traffic represents outbound traffic except for availability monitoring and webhooks, which require inbound firewall rules.

You need to open some **outgoing ports** in your server's firewall to allow the Application Insights SDK and/or Status Monitor to send data to the portal:

Expand table

Purpose	URL	IP address	Ports
Telemetry	dc.applicationinsights.us	23.97.4.113	443

Cost Management and Billing

The following Azure Cost Management + Billing **features aren't currently available** in Azure Government:

- Cost Management + Billing for cloud solution providers (CSPs)

Media

This section outlines variations and considerations when using Media services in the Azure Government environment. For service availability, see Products available by region.

Media Services

For Azure Media Services v3 feature variations in Azure Government, see Azure Media Services v3 clouds and regions availability.

Migration

This section outlines variations and considerations when using Migration services in the Azure Government environment. For service availability, see Products available by region.

Azure Migrate

The following Azure Migrate **features aren't currently available** in Azure Government:

- Containerizing Java Web Apps on Apache Tomcat (on Linux servers) and deploying them on Linux containers on App Service.
- Containerizing Java Web Apps on Apache Tomcat (on Linux servers) and deploying them on Linux containers on Azure Kubernetes Service (AKS).
- Containerizing ASP.NET apps and deploying them on Windows containers on AKS.
- Containerizing ASP.NET apps and deploying them on Windows containers on App Service.
- You can only create assessments for Azure Government as target regions and using Azure Government offers.

For more information, see Azure Migrate support matrix. For a list of Azure Government URLs needed by the Azure Migrate appliance when connecting to the internet, see Azure Migrate appliance URL access.

Networking

This section outlines variations and considerations when using Networking services in the Azure Government environment. For service availability, see Products available by region.

Azure ExpressRoute

For an overview of ExpressRoute, see What is Azure ExpressRoute?. For an overview of how **BGP communities** are used with ExpressRoute in Azure Government, see BGP community support in National Clouds.

Azure Front Door

Azure Front Door (AFD) Standard and Premium tiers are available in general availability in Azure Government regions US Gov Arizona and US Gov Texas. The following Azure Front Door feature **isn't supported** in Azure Government:

- Managed certificate for enabling HTTPS; instead use your own certificate.

Private Link

- For Private Link services availability, see Azure Private Link availability.
- For Private DNS zone names, see Azure Private Endpoint DNS configuration.

Traffic Manager

Traffic Manager health checks can originate from certain IP addresses for Azure Government. Review the IP addresses in the JSON file to ensure that incoming connections from these IP addresses are allowed at the endpoints to check its health status.

Security

This section outlines variations and considerations when using Security services in the Azure Government environment. For service availability, see Products available by region.

Microsoft Defender for Endpoint

For feature variations and limitations, see Microsoft Defender for Endpoint for US Government customers.

Microsoft Defender for IoT

For feature variations and limitations, see Cloud feature availability for US Government customers.

Azure Information Protection

Azure Information Protection Premium is part of the Enterprise Mobility + Security suite. For details on this service and how to use it, see Azure Information Protection Premium Government Service Description.

Microsoft Defender for Cloud

For feature variations and limitations, see Cloud feature availability for US Government customers.

Microsoft Sentinel

For feature variations and limitations, see Cloud feature availability for US Government customers.

Storage

This section outlines variations and considerations when using Storage services in the Azure Government environment. For service availability, see Products available by region.

Azure NetApp Files

For Azure NetApp Files feature availability in Azure Government and how to access the Azure NetApp Files service within Azure Government, see Azure NetApp Files for Azure Government.

Azure Import/Export

With Import/Export jobs for US Gov Arizona or US Gov Texas, the mailing address is for US Gov Virginia. The data is loaded into selected storage accounts from the US Gov Virginia region. For all jobs, we recommend that you rotate your storage account keys after the job is complete to remove any access granted during the process. For more information, see Manage storage account access keys.

Web

This section outlines variations and considerations when using Web services in the Azure Government environment. For service availability, see Products available by region.

API Management

The following API Management **features aren't currently available** in Azure Government:

- Azure AD B2C integration

App Service

The following App Service **resources aren't currently available** in Azure Government:

- App Service Certificate
- App Service Managed Certificate
- App Service Domain

The following App Service **features aren't currently available** in Azure Government:

- Deployment
  - Deployment options: only Local Git Repository and External Repository are available

Azure Functions

When connecting your Functions app to Application Insights in Azure Government, make sure you use APPLICATIONINSIGHTS_CONNECTION_STRING, which lets you customize the Application Insights endpoint.

Next steps

Learn more about Azure Government:

- Acquiring and accessing Azure Government
- Azure Government overview
- Azure support for export controls
- Azure Government compliance
- Azure Government security
- Azure guidance for secure isolation

Start using Azure Government:

- Guidance for developers
- Connect with the Azure Government portal