AWS Panorama

Developer Guide

AWS Panorama: Developer Guide

Table of Contents

What is AWS Panorama?

With AWS Panorama, you can build computer vision applications for your business or customers without purchasing special cameras. By using the AWS Panorama Appliance with your existing network cameras, you can run applications that use machine learning (ML) to collect data from video streams, output video with text and graphical overlays, and interact with other AWS services.

The AWS Panorama Appliance is a compact edge appliance that uses a powerful system-on-module (SOM) that is optimized for machine learning workloads. The appliance can run multiple computer vision models against multiple video streams in parallel and output the results in real time with limited internet connectivity. It is designed for use in commercial and industrial settings and is rated for dust and liquid protection (IP-62).

> **Note**
> AWS Panorama is available in preview with the AWS Panorama Appliance Developer Kit for developing computer vision applications in a non-production environment. The AWS Panorama Appliance will be available at AWS Panorama GA. To sign up for preview and get an AWS Panorama Appliance Developer Kit, visit aws.amazon.com/panorama.

The AWS Panorama Appliance enables you to run self-contained computer vision applications at the edge, without sending images to the AWS Cloud. By using the AWS SDK, you can integrate with other AWS services and use them to track data from the application over time. By integrating with other AWS services, you can use AWS Panorama to do the following:

- **Improve quality control** – Monitor an assembly line's output to identify parts that don't conform to requirements. Highlight images of nonconformant parts with text and a bounding box and display them on a monitor for review by your quality control team.
- **Analyze traffic patterns** – Use the AWS SDK to record data for retail analytics in Amazon DynamoDB. Use a serverless application to analyze the collected data over time, detect anomalies in the data, and predict future behavior.
- **Receive site safety alerts** – Monitor off-limits areas at an industrial site. When your application detects a potentially unsafe situation, upload an image to Amazon Simple Storage Service (Amazon S3) and send a notification to an Amazon Simple Notification Service (Amazon SNS) topic so recipients can take corrective action.
- **Collect training and test data** – Upload images of objects that your computer vision model couldn't identify, or where the model's confidence in its guess was borderline. Use a serverless application to create a queue of images that need to be tagged. Tag the images and use them to retrain the model in Amazon SageMaker.

AWS Panorama uses other AWS services to manage the AWS Panorama Appliance, access models and code, and deploy applications. AWS Panorama does as much as possible without requiring you to interact with other services, but a knowledge of the following services can help you understand how AWS Panorama works.

- SageMaker – You can use SageMaker to collect training data from cameras or sensors, build a machine learning model, and train it for computer vision. AWS Panorama can import trained computer vision models from SageMaker, and it uses SageMaker Neo to optimize models to run on the AWS Panorama Appliance.
- Amazon S3 – You use Amazon S3 to store trained models, even those built with an ML framework other than SageMaker, for use with AWS Panorama. You can also use Amazon S3 to store your application code.

- AWS Lambda – You author AWS Panorama application code in Lambda. Although your code doesn't run in Lambda, you can use it for versioning and to store the application code that AWS Panorama deploys to the AWS Panorama Appliance.
- AWS IoT – AWS Panorama uses AWS IoT services to monitor the state of the AWS Panorama Appliance, manage software updates, and deploy applications. You don't need to use AWS IoT directly.

To get started with the AWS Panorama Appliance and learn more about the service, continue to Getting started with AWS Panorama (p. 3).

Getting started with AWS Panorama

To get started with AWS Panorama, first learn about the service's concepts (p. 4) and the terminology used in this guide. Then you can use the AWS Panorama console to register your AWS Panorama Appliance Developer Kit (p. 5) and create an application (p. 8). In about an hour, you can configure the device, update its software, and deploy a sample application. To complete the tutorials in this section, you use the AWS Panorama Appliance Developer Kit and a camera that streams video over a local network.

The AWS Panorama sample application (p. 12) analyzes a video stream to tally the number of people detected and display the results on a connected display. It includes a model that has been trained with SageMaker and sample code that uses the AWS Panorama Application SDK to run inference and output video.

After you've used the developer kit with the sample application, connect to the AWS Panorama Appliance Developer Kit (p. 16) with SSH. When you are logged in to the appliance, you can explore the execution environment and view logs for applications, cameras, and the appliance itself.

The final two topics in this chapter detail requirements for models and cameras (p. 20), and the hardware specifications of the AWS Panorama Appliance Developer Kit (p. 21). If you haven't obtained an appliance and cameras yet, or plan on developing your own computer vision models, see these topics first for more information.

Topics

AWS Panorama concepts

In AWS Panorama, you create computer vision applications and deploy them to the AWS Panorama Appliance to analyze video streams from network cameras. You write application code in Python and import machine learning models from Amazon SageMaker or Amazon Simple Storage Service (Amazon S3). Applications use the AWS Panorama Application SDK to receive video input from a datasource and output it to a data sink.

Concepts

The AWS Panorama Appliance

The AWS Panorama Appliance is the hardware that runs your applications. You use the AWS Panorama console to register an appliance, update its software, and deploy applications to it. The software on the AWS Panorama Appliance discovers and connects to camera streams, sends frames of video to your application, and displays video output on an attached display.

The AWS Panorama Appliance is an *edge device*. Instead of sending images to the AWS Cloud for processing, it runs applications locally on optimized hardware. This enables you to analyze video in real time and process the results with limited connectivity. The appliance requires an internet connection to report its status, to upload logs, and to get software updates and deployments.

For more information, see Managing the AWS Panorama Appliance (p. 26).

Applications

Applications run on the AWS Panorama Appliance to perform computer vision tasks on video streams. You can build computer vision applications by combining Python code and machine learning models, and deploy them to the AWS Panorama Appliance over the internet. Applications can send video to a display, or use the AWS SDK to send results to AWS services.

For more information, see Building AWS Panorama applications (p. 34).

Models

A computer vision model is a machine learning network that is trained to process images. Computer vision models can perform various tasks such as classification, detection, segmentation, and tracking. A computer vision model takes an image as input and outputs information about the image or objects in the image.

AWS Panorama supports models built with PyTorch, Apache MXNet, and TensorFlow. You can build models with Amazon SageMaker and import them from a SageMaker job or an Amazon Simple Storage Service (Amazon S3) bucket. For more information, see ??? (p. 37).

Setting up the AWS Panorama Appliance Developer Kit

To get started using your AWS Panorama Appliance Developer Kit, register it in the AWS Panorama console and update its software. During the setup process, you create an appliance *resource* in AWS Panorama that represents the physical appliance, and copy files to the appliance with a USB drive. The appliance uses these certificates and configuration files to connect to the AWS Panorama service. Then you use the AWS Panorama console to update the appliance's software and connect cameras.

Sections

Prerequisites

To follow this tutorial, you need an AWS Panorama Appliance Developer Kit and the following hardware:

- **Display** – A display with HDMI input for viewing the sample application output
- **USB drive** (included) – A FAT32-formatted USB flash memory drive with at least 1 GB of storage, for transferring an archive with configuration files and a certificate to the AWS Panorama Appliance Developer Kit
- **Camera** – An IP camera (p. 20) that outputs an RTSP video stream for providing input to the camera. The developer kit can automatically discover streams from cameras that support ONVIF Profile S.

The tutorial uses a sample computer vision model and application code. Download the model and code before you get started.

- **Model** – ssd_512_resnet50_v1_voc.tar.gz
- **Code** – aws-panorama-sample.zip

The AWS Panorama console uses other AWS services to assemble application components, manage permissions, and verify settings. To register a developer kit and deploy the sample application, you need access to the following services:

- **Amazon Simple Storage Service (Amazon S3)** – To store model and Lambda function artifacts, and can be used for application output.
- **AWS Lambda** – To manage function code, configuration, and versions.
- **AWS Identity and Access Management (IAM)** – On first run, to create roles used by the AWS Panorama service, the AWS Panorama console, the AWS Panorama Appliance Developer Kit, AWS IoT Greengrass, SageMaker, and Lambda functions.

If you don't have permission to create roles in IAM, have an administrator open the AWS Panorama console and accept the prompt to create service roles. For the Lambda function's permissions, you can create an execution role with basic permissions ahead of time, in which case you only need permission to pass the role.

Register and configure the developer kit

The AWS Panorama Appliance is a hardware device that connects to network-enabled cameras over a local network connection. It uses a Linux-based operating system that includes the AWS Panorama Application SDK and supporting software for running computer vision applications.

To connect to AWS for appliance management and application deployment, the AWS Panorama Appliance Developer Kit uses device certificates. You use the AWS Panorama console to generate certificates that authenticate the developer kit and authorize it to call AWS API operations.

When you set up the AWS Panorama Appliance Developer Kit, enable SSH to so that you can connect to it for testing and debugging. To enable Wi-Fi, configure an SSID and password during setup. Enabling Wi-Fi does not disable Ethernet, and a wired connection is required to complete setup prior to updating the appliance's software.

To register an AWS Panorama Appliance Developer Kit

1. Open the AWS Panorama console Getting started page.
2. Choose **Set up appliance**.
3. Follow the instructions to create the appliance resource, configure network access, and download an archive with the device certificate and configuration files.
4. Copy the configuration archive to the root directory of the USB drive.
5. Connect the USB drive to the developer kit and turn it on.
6. The developer kit copies the configuration archive and network configuration file to itself, connects to the network, and connects to the AWS Cloud. To continue, choose **Next**.
7. Do not add cameras at this time. Proceed through the remaining steps to complete setup.

Upgrade the developer kit software

The AWS Panorama Appliance Developer Kit has several software components, including a Linux operating system, the AWS Panorama application SDK (p. 45), and supporting computer vision libraries and frameworks. To ensure that you can use the latest features and applications with your developer kit, upgrade its software after setup and whenever an update is available.

To update the appliance software

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose **Settings**
4. Under **System software**, choose **Install version**.

> **Important**
> Before you continue, remove the USB drive from the developer kit and format it to delete its contents. The configuration archive contains sensitive data and is not deleted automatically.

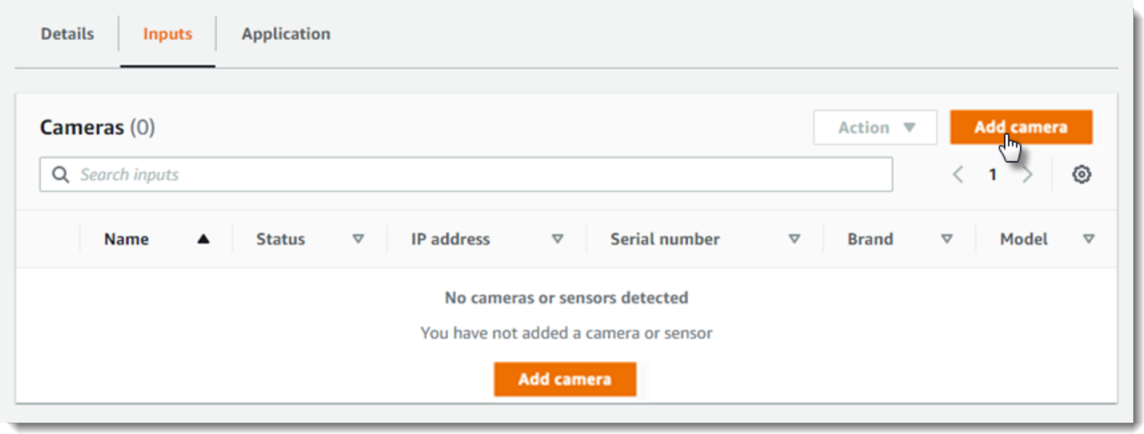
The upgrade process can take 30 minutes or more.

Add a camera stream

After the software upgrade completes, add a camera stream.

To add a camera stream to the AWS Panorama Appliance

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose **Inputs**.
4. Choose **Add camera**.



Name	Status	IP address	Serial number	Brand	Model

No cameras or sensors detected

You have not added a camera or sensor

5. Choose a connection mode. Try **Automatic** first. If it doesn't find your camera stream, use **Manual**.

   - **Automatic** – The AWS Panorama Appliance discovers cameras on the local network. Choose a camera and then choose a stream to add. If the camera has multiple streams, repeat the process to add additional streams.
   - **Manual** – Enter the IP address of the camera and the RTSP URL of a stream.

   Both workflows support password-protected cameras.
6. Choose **Confirm**.

Next steps

If you encountered errors during setup, see Troubleshooting (p. 49).

To deploy a sample application, continue to the next topic (p. 8).

Deploying the AWS Panorama sample application

After you've set up your AWS Panorama Appliance Developer Kit (p. 5) and upgraded its software, deploy a sample application. In the following sections, you define the application code, import a machine learning model, and deploy an application with the AWS Panorama console.

The sample application uses a machine learning model to detect people in frames of video from a network camera. It uses the AWS Panorama Application SDK to load a model, get images, and run the model. The application then overlays the results on top of the original video and outputs it to a connected display.

In a retail setting, analyzing foot traffic patterns enables you to predict traffic levels. By combining the analysis with other data, you can plan for increased staffing needs around holidays and other events, measure the effectiveness of advertisements and sales promotions, or optimize display placement and inventory management.

Sections

Create a bucket for storage

Create an Amazon S3 bucket for the sample model file.

To create a bucket

1. Open the Amazon S3 console.
2. Choose **Create bucket**.
3. Follow the instructions to create a bucket with the following settings:
   - **Bucket name** – The name must contain the phrase `aws-panorama`. For example, `aws-panorama-artifacts-123456789012`.
   - **Region** – The AWS Region where you use AWS Panorama.
4. Upload the model archive (p. 5) to the bucket.

When the upload operation is complete, view the details of the model file and note the **S3 URL** value for later use.

Create a Lambda function

Use the Lambda console to create the Lambda function that the application uses to run inference against the model. The code is stored and versioned in Lambda, but it is not invoked in Lambda. You deploy the code to the AWS Panorama Appliance Developer Kit and it runs continually while the developer kit is on.

When you create a Lambda function, you give it a role that enables it to upload logs and access services with the AWS SDK. This role is typically used when the function runs in Lambda, but when you run the application on the developer kit, the developer kit's permissions are used. You can let the Lambda console create a role or, if you don't have permission to create roles, use a role that an administrator creates for you.

To create a Lambda function

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose **Create function**.
3. For **Basic information**, configure the following settings:
   - **Function name** – `aws-panorama-sample-function`.
   - **Runtime** – **Python 3.7**.
   - **Permissions** – To create a new role, use the default setting. If you have a role that you want to use, choose **Use an existing role**.
4. Choose **Create function**.

Lambda creates the function and opens the function details page. Modify the function code and configure the function's runtime settings. Then, to create an immutable snapshot of the function's code and configuration, publish a version.

To update the function's code and configuration

1. For **Code source**, choose **Upload from** and then choose **.zip file**.
2. Upload the sample code deployment package and then choose **Save**.
3. When the operation is complete, in the **Runtime settings** section, choose **Edit**.
4. Configure the following settings:
   - **Handler** – `lambda_function.main`. In the sample application, the `lambda_function.py` file exports a method named `main` that serves as the handler.
5. Choose **Save**.
6. Above the code editor, choose the **Configuration** tab.
7. In the **General configuration** section, choose **Edit**.
8. Configure the following settings:
   - **Timeout** – **2 minutes**.
   - **Memory** – **2048 MB**.
9. Choose **Save**.
10. Choose **Actions**.
11. Choose **Publish new version**, and then choose **Publish**.

When you configure an application, you choose a function version. Using a version ensures that the application continues to work if you make changes to the function's code. To update the application's code, you publish a new version in Lambda and configure the application to use it.

Create the application

In this example, the application uses a Lambda function named `aws-panorama-sample-function` to run inference against the aws-panorama-sample-model model on video streams from a camera. The function displays the result on an HDMI display connected to the developer kit.

To import the sample model and create an application, use the AWS Panorama console.

To create an application

1. Open the AWS Panorama console Getting started page.
2. Choose **Create application**.
3. Complete the workflow with the following settings:

   - **Name** – **`aws-panorama-sample`**
   - Model source – **External model**
   - **Model artifact path** – The Amazon S3 URI of the model archive (p. 5). For example: **`s3://aws-panorama-artifacts-123456789012/ssd_512_resnet50_v1_voc.tar.gz`**
   - **Model name** – **`aws-panorama-sample-model`**

     This value is used by the application code. Enter it exactly as shown.
   - **Model framework** – **MXNet**
   - **Input name** – **`data`**
   - **Shape** – **`1,3,512,512`**
   - **Lambda functions** – aws-panorama-sample-function version 1

The value for **Shape**, `1,3,512,512`, indicates the number of images that the model takes as input (1), the number of channels in each image (3--red, green, and blue), and the dimensions of the image (512 x 512). The values and order of the array varies among models.

Deploy the application

Use the AWS Panorama console to deploy the application to your AWS Panorama Appliance Developer Kit. AWS Panorama uses AWS IoT Greengrass to deploy the application code and model to the developer kit.

To deploy the application

1. Open the AWS Panorama console Applications page.
2. To open the application page, choose **aws-panorama-sample**.
3. Choose **Deploy**.
4. Follow the instructions to deploy the application.

When the deployment is complete, the application starts processing the video stream and displays the output on the connected monitor.

If the application doesn't start running, check the application and device logs (p. 48) in Amazon CloudWatch Logs.

Clean up

If you are done working with the sample application, you can use the AWS Panorama console to remove it from the developer kit and delete it.

To remove the application from the developer kit

1. Open the AWS Panorama console Appliances page.
2. Choose the developer kit.
3. Check the box next to the application's name.
4. Choose **Delete application**.

To delete the application from AWS Panorama

1. Open the AWS Panorama console Applications page.
2. Choose an application.
3. Choose **Delete**.

The Lambda function and Amazon S3 bucket that you created are not deleted automatically. You can delete them in the Lambda console and Amazon S3 console, respectively.

Next steps

If you encountered errors while deploying or running the sample application, see Troubleshooting (p. 49).

To learn more about the sample application's features and implementation, continue to the next topic (p. 16).

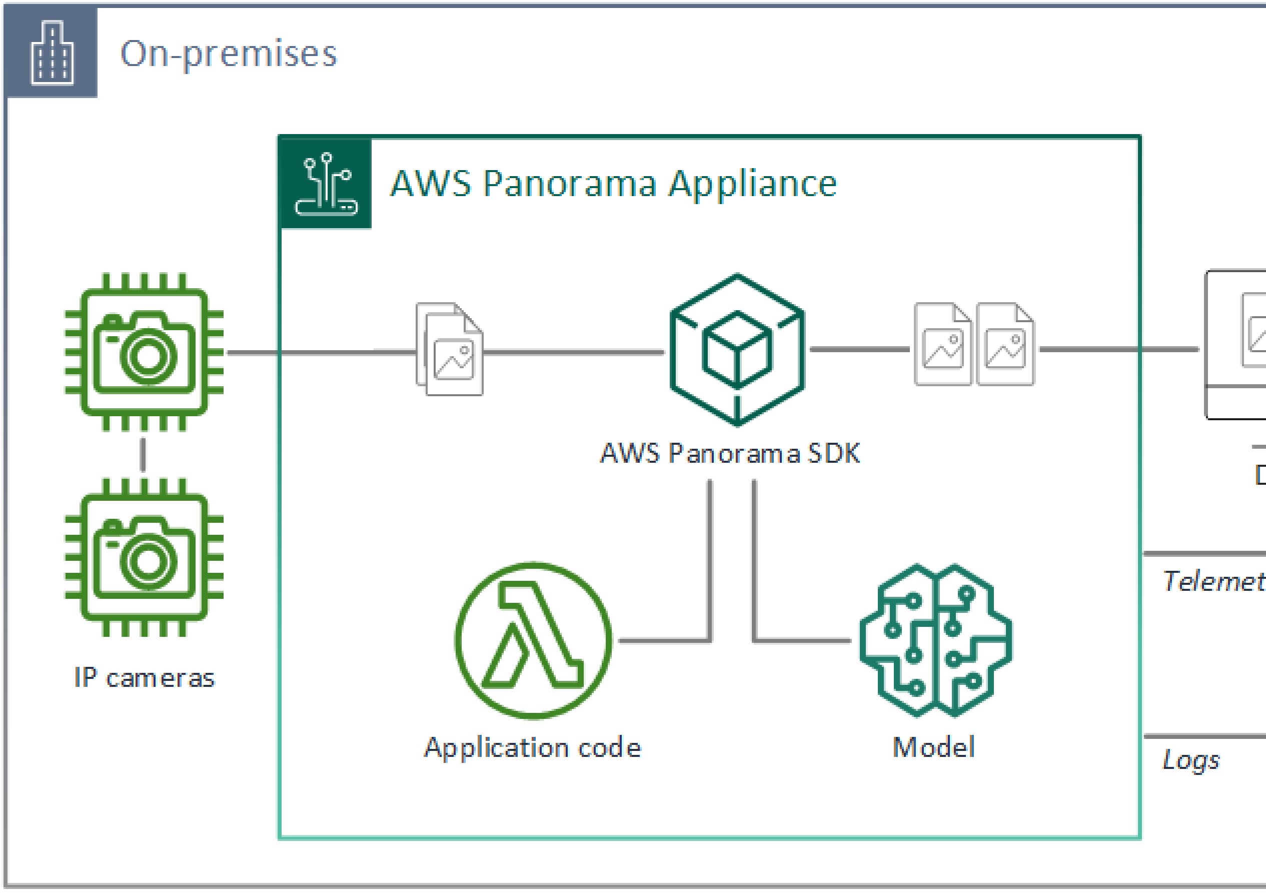
To learn about the AWS Panorama Appliance Developer Kit, continue to Using the AWS Panorama Appliance Developer Kit (p. 16).

AWS Panorama sample application features

A sample application that demonstrates the use of the AWS Panorama Application SDK is available in this guide's GitHub repository. You can run the sample application by following the previous topic (p. 8) or by using the command-line scripts and instructions in the project's README file.

The sample application uses a computer vision model to detect people in single frames of video. When it detects one or more people, it draws an overlay on the image that indicates where the people are and how many people are detected. The application can process multiple video streams. It outputs a video stream that shows the original video or videos with overlays on top.

The following diagram shows the major components of the application running on an AWS Panorama Appliance Developer Kit. The application code uses the AWS Panorama Application SDK to get images and interact with the model, which it doesn't have direct access to. The application outputs video to a connected display but does not send image data outside of your local network.



To simplify deploying and managing the application, the project includes a AWS CloudFormation template that uses AWS Serverless Application Model (AWS SAM) to create an AWS Lambda function and its runtime role. It also includes shell scripts that create a bucket, upload the model, deploy the application, and clean up everything when you're done.

Note
For more sample code, see the aws-panorama-samples repository on GitHub.

The following sections describe features of the sample application and highlight relevant selections of code. The code examples are abbreviated for clarity. For the full application code, see the linked source files in GitHub.

Sections

Loading a model

During initialization, the application uses the AWS Panorama Application SDK to load a machine learning model. First, it creates a model object by calling `panoramasdk.model()`. Then, it uses the `open` method to load the model by name.

Example lambda_function.py – Initialization

```
    def init(self, parameters, inputs, outputs):
        try:
            self.threshold = parameters.threshold
            self.person_index = parameters.person_index
            self.frame_num = 0
            ...
            # Load model
            logger.info("Loading model: " + parameters.model_name)
            self.model = panoramasdk.model()
            self.model.open(parameters.model_name, 1)
            # Use 16-bit precision for faster inference
            os.environ['TVM_TENSORRT_USE_FP16'] = '1'
            # Create input and output arrays
            class_info = self.model.get_output(0)
            prob_info = self.model.get_output(1)
            rect_info = self.model.get_output(2)
            self.class_array = np.empty(class_info.get_dims(), dtype=class_info.get_type())
            self.prob_array = np.empty(prob_info.get_dims(), dtype=prob_info.get_type())
            self.rect_array = np.empty(rect_info.get_dims(), dtype=rect_info.get_type())
```

When you deploy the application, you create the model resource in the AWS Panorama console and name it `aws-panorama-sample-model`. This is the default value for the `model_name` parameter. To increase inference speed, the TVM_TENSORRT_USE_FP16 configures the appliance to use 16-bit floating-point numbers instead of 32-bit. You can omit this setting if your model performs well without it.

Detecting objects

The application starts by calling the `run` method that it inherits from `awspanoramasdk.base`. This method calls the `entry` method and passes it frames of video from one or more camera streams (`video_in`). The application loops over frames of video, detects people, and outputs the original video with an overlay that includes the number of people detected, and bounding boxes around each person.

Example lambda_function.py – Inference

```
class people_counter(panoramasdk.base):
    ...
    def entry(self, inputs, outputs):
        self.frame_num += 1
        # Loop through attached video streams
        for i in range(len(inputs.video_in)):
            outputs.video_out[i] = self.process_media(inputs.video_in[i])
        return True

    def process_media(self, media):
        stream = media.stream_uri
        # Set up stream buffer
        if not self.buffered_media.get(stream):
            self.buffered_media[stream] = media
            self.buffered_image[stream] = self.preprocess(media.image)
            logger.info('Set up frame buffer for stream: {}'.format(stream))
            logger.info('Stream image size: {}'.format(media.image.shape))
        output = self.buffered_media[stream]
        # Run inference on the buffered image
        self.model.batch(0, self.buffered_image[stream])
        self.model.flush()
        # While waiting for inference, preprocess the current image
        self.buffered_image[stream] = self.preprocess(media.image)
        self.buffered_media[stream] = media
        # Wait for inference results
        inference_results = self.model.get_result()
        # Process results
        output = self.process_results(inference_results, output)
        self.model.release_result(inference_results)
        return output
...
def main():
    people_counter().run()

main()
```

The application stores the latest frame of video from each stream in a buffer. While it runs inference to detect objects on the previous image, the application prepares the new image for inference in parallel. Between calls to `model.flush()` and `model.get_result()`, the model is running inference asynchronously on the appliance's GPU. During this time, the application can use the CPU to preprocess images, reducing the total processing time for a frame by about 15 ms.

Preprocessing images

Before the application sends an image to the model, it prepares it for inference by resizing it and normalizing color data. The model that the application uses requires a 512 x 512 pixel image with three color channels, to match the number of inputs in its first layer. The application adjusts each color value by converting it to a number between 0 and 1, subtracting the average value for that color, and dividing by the standard deviation. Finally, it combines the color channels and converts it to a NumPy array that the model can process.

Example lambda_function.py – Preprocessing

```
    def preprocess(self, img):
        resized = cv2.resize(img, (HEIGHT, WIDTH))
        mean = [0.485, 0.456, 0.406]
        std = [0.229, 0.224, 0.225]
        img = resized.astype(np.float32) / 255.
        img_a = img[:, :, 0]
        img_b = img[:, :, 1]
        img_c = img[:, :, 2]
        # Normalize data in each channel
        img_a = (img_a - mean[0]) / std[0]
        img_b = (img_b - mean[1]) / std[1]
        img_c = (img_c - mean[2]) / std[2]
```

```
       # Put the channels back together
       x1 = [[[], [], []]]
       x1[0][0] = img_a
       x1[0][1] = img_b
       x1[0][2] = img_c
       return np.asarray(x1)
```

This process gives the model values in a predictable range centered around 0. It matches the preprocessing applied to images in the training dataset, which is a standard approach but can vary per model.

Processing results

When the application runs inference, it sends the model only a single image at a time. The AWS Panorama Application SDK can also process images in batches, so it takes a batch as input and sends results in batches. The *batch set* contains a batch of results for each output from the model.

The sample model returns three outputs: a list of objects detected, a list of probabilities for those objects, and a list of bounding boxes. The application gets the batch for each output, and then the list for the first image in each batch. It finds people among those objects, and updates the output image accordingly.

Example lambda_function.py – Process results

```
   def process_results(self, batch_set, output_media):
       # Model outputs (classes, probabilities, bounding boxes) are collected in
       # the BatchSet returned by model.get_result
       # Each output is a Batch of arrays, one for each input in the batch
       classes = batch_set.get(0)
       probabilities = batch_set.get(1)
       boxes = batch_set.get(2)
       # Each batch has only one image; save results for that image
       classes.get(0, self.class_array)
       probabilities.get(0, self.prob_array)
       boxes.get(0, self.rect_array)
       # Get indices of people in class array
       person_indices = [i for i in range(len(self.class_array[0])) if
int(self.class_array[0][i]) == self.person_index]
       # Filter out results beneath confidence threshold
       prob_person_indices = [i for i in person_indices if self.prob_array[0][i] >=
self.threshold]
       # Draw bounding boxes on output image
       for index in prob_person_indices:
           left = np.clip(self.rect_array[0][index][0] / np.float(HEIGHT), 0, 1)
           top = np.clip(self.rect_array[0][index][1] / np.float(WIDTH), 0, 1)
           right = np.clip(self.rect_array[0][index][2] / np.float(HEIGHT), 0, 1)
           bottom = np.clip(self.rect_array[0][index][3] / np.float(WIDTH), 0, 1)
           output_media.add_rect(left, top, right, bottom)
           output_media.add_label(str(self.prob_array[0][index][0]), right, bottom)
       # Add text
       output_media.add_label('People detected: {}'.format(len(prob_person_indices)),
0.02, 0.9)
       return output_media
```

Using the AWS Panorama Appliance Developer Kit

The AWS Panorama Appliance Developer Kit is an appliance for developing and testing AWS Panorama applications. You connect to the developer kit from your computer to run commands, view logs, and explore the AWS Panorama Application SDK.

> **Important**
> The AWS Panorama Appliance Developer Kit is not secured for use on production networks or workloads. For more information, see Security considerations for the AWS Panorama Appliance Developer Kit (p. 64).

This tutorial provides an introduction to connecting to the developer kit with SSH, viewing logs, and using Python libraries in application code.

Steps

- Prerequisites (p. 16)
- Connect with SSH (p. 16)
- Get superuser access (p. 17)
- View logs (p. 17)
- View the AWS Panorama Application SDK help (p. 18)
- Use the AWS SDK for Python (Boto3) (p. 19)
- Next steps (p. 19)

Prerequisites

To connect to the developer kit and run commands, you must enable SSH during setup (p. 5).

> **Important**
> Be sure to record the username and password for SSH. They are not stored in AWS Panorama. If you lose them, you must repeat the setup process.

To follow the procedures in this tutorial, you need a command line terminal or shell to run commands. In the code listings, commands are preceded by a prompt symbol ($) and the name of the current directory, when appropriate.

```
~/lambda-project$ this is a command
this is output
```

For long commands, we use an escape character (\) to split a command over multiple lines.

On Linux and macOS, use your preferred shell and package manager. On Windows 10, you can install the Windows Subsystem for Linux to get a Windows-integrated version of Ubuntu and Bash.

This tutorial includes sample scripts that you can run on the developer kit as-is or by replacing some values with your own. For scripts and other resources that you want to save, you can create a directory under `/data` on the developer kit's local storage. Files outside of `/data` are not saved when you update your developer kit's software.

Connect with SSH

The AWS Panorama Appliance Developer Kit reports its IP address to AWS Panorama. You can find it in the AWS Panorama console, on the developer kit's settings page. Use this address to connect to the developer kit with SSH.

To get the developer kit's IP address

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose **Settings**.
4. Find the IP address under **Appliance network**.

> **Note**
> The developer kit renews its IP address lease with your network's DHCP server automatically. If you disconnect the developer kit from your network for longer than the lease time. it might be assigned a different IP address. To avoid this, you can configure a static IP address during the setup process (p. 5).

To connect to the developer kit, use the ssh command with your username and the developer kit's IP address.

```
$ ssh 10.24.34.0 -l me
me@10.24.34.0's password:
```

To end the session and return to your computer's shell, use the exit command.

To copy logs and other files from the developer kit onto your computer, you can use the scp command. The following example copies the AWS IoT job agent log from a developer kit to the current directory.

```
$ scp me@10.0.0.100:/opt/aws/panorama/iot/jobs_agent.log .
me@10.0.0.100's password:
jobs_agent.log                                              100% 9049KB 372.4KB/s   00:24
```

Get superuser access

To use some commands, you need to elevate your privileges. To assume superuser privileges for a single command, use sudo. For example, to view a protected log file, you can use the sudo command with tail.

```
me@tegra-ubuntu$ sudo tail /var/log/awslogs.log
```

The first time you use sudo, you must enter your user password. You can continue to run commands with sudo for a few minutes without entering your password again. To avoid entering your password multiple times in a session, assume the privileges of the root user.

```
me@tegra-ubuntu$ sudo -su root
root@tegra-ubuntu:/home/me# tail /var/log/awslogs.log
```

> **Important**
> Be careful when using root privileges. Making changes to the software or file system can have unintended consequences. When you update the developer kit, change settings in the AWS Panorama console, or deploy applications, changes to the developer kit can be reverted.

To exit superuser mode, use the exit command.

View logs

AWS Panorama generates logs in several locations on the developer kit. In addition to logs for your application code,AWS Panorama generates logs for camera streams, outputs, and AWS client processes. The AWS Panorama Appliance sends many of these logs to Amazon CloudWatch Logs.

Your application code's output log is stored under `/data/greengrass`, in a folder structure that includes your AWS Region, account number, and AWS Lambda function name. To use the following example, replace the account ID and the name of the Lambda function with your own.

Example get-app-log.sh (requires sudo) – Show application log

```
REGION=us-east-1
ACCOUNT=123456789012
FUNCTION=aws-panorama-sample-function
LOG=/data/greengrass/ggc/var/log/user/$REGION/$ACCOUNT/$FUNCTION.log
tail -100 $LOG
```

In addition to application logs, you can view the system logs for AWS IoT Greengrass.

Example list-greengrass-logs.sh (requires sudo) – List AWS IoT and AWS IoT Greengrass logs

```
PANORAMA=/opt/aws/panorama
IOT=$PANORAMA/iot
GREENGRASS=/data/greengrass/ggc/var/log/system
echo $IOT:
ls -gGht $IOT
echo ""
echo $GREENGRASS:
ls -gGht $GREENGRASS
```

To view logs for cameras (datasources) and HDMI output (datasinks), navigate to the `mediapipeline` directory under `/opt/aws/panorama`. Each camera and display has a separate log file.

Example list-mediapipeline-logs.sh – Get a list of camera and display logs

```
PANORAMA=/opt/aws/panorama
PIPELINE=$PANORAMA/mediapipeline/bin
SOURCES=$PIPELINE/source_onvif-arn-aws-panorama-
SINKS=$PIPELINE/sink_hdmi-datasink-hdmi-
ls $SOURCES*
ls $SINKS*
```

The Amazon CloudWatch Logs agent outputs logs to `/var/log/awslogs.log`. If you don't see logs for your developer kit in CloudWatch Logs, check this file for errors.

Example get-cwl-log.sh (requires sudo) – Show CloudWatch Logs agent log

```
CWL=/var/log/awslogs.log
tail $CWL
```

For more information about using CloudWatch Logs to view logs, see Viewing AWS Panorama event logs in CloudWatch Logs (p. 48).

View the AWS Panorama Application SDK help

The AWS Panorama Application SDK is a Python library that is included in the developer kit's software image. You use it in your application code to interact with the developer kit and model.

To see information about the library, load it into a Python interpreter and use the `help` command.

```
$ python3
Python 3.7.5
```

```
>>> import panoramasdk
>>> help(panoramasdk)

    CLASSES
        builtins.list(builtins.object)
            port
        builtins.object
            base
            batch
            batch_set
    ...
```

The documentation for each class shows details about its methods' behavior and parameters.

For more information about the SDK, see The AWS Panorama Application SDK (p. 45).

Use the AWS SDK for Python (Boto3)

To access resources in other AWS services, you can use the SDK for Python in your application code. When your application code runs, it gets permission to call AWS services from an AWS Identity and Access Management (IAM) role that is attached to the device.

To call AWS services with the SDK for Python, import the library (`boto3`) and create a client. The following example shows how to use an Amazon Simple Storage Service (Amazon S3) client to get a list of objects in an AWS Panorama artifacts bucket.

```
$ python3
Python 3.7.5

>>> import boto3
>>> s3 = boto3.client('s3')
>>> bucket = 'aws-panorama-artifacts-123456789012'
>>> response = s3.list_objects(Bucket=bucket)
>>> for object in response['Contents']:
...     print(f'    {object["Key"]}')
...
ssd_512_resnet50_v1_voc.tar.gz
```

The AWS Panorama Appliance has limited permissions to access objects in buckets that include `aws-panorama` in the name, and a few other resources. You can add permissions to the developer kit's service role (AWSPanoramaGreengrassGroupRole (p. 25)).

Next steps

If you encountered errors while connecting to the developer kit or running commands, see Troubleshooting (p. 49).

Next, learn more about AWS Panorama concepts (p. 4).

Supported computer vision models and cameras

AWS Panorama integrates with Amazon SageMaker to support importing and compiling computer vision models for AWS Panorama applications. You can start with a provided model, or build your own model with a supported framework. You can import a model from an Amazon Simple Storage Service (Amazon S3) bucket or from the output of a SageMaker job.

To process video and get images to send to a model, the AWS Panorama Appliance connects to an H.264 encoded video stream with the RTSP protocol. AWS Panorama tests a variety of common cameras for compatibility.

Sections

Supported models

When you build an application for AWS Panorama, you provide a machine learning model that the application uses for computer vision. You can use pre-built and pre-trained models provided by model frameworks, a sample model (p. 37), or a model that you build and train yourself.

> **Note**
> For a list of pre-built models that have been tested with AWS Panorama, see Model compatibility.

When you deploy an application, AWS Panorama uses the SageMaker Neo compiler to compile your computer vision model. SageMaker Neo is a compiler that optimizes models to run efficiently on a target platform, which can be an instance in Amazon Elastic Compute Cloud (Amazon EC2), or an edge device such as the AWS Panorama Appliance.

AWS Panorama supports the versions of PyTorch, Apache MXNet, and TensorFlow that are supported for edge devices by SageMaker Neo. When you build your own model, you can use the framework versions listed in the SageMaker Neo release notes.

For more information about using models in AWS Panorama, see Computer vision models (p. 37).

Supported cameras

The AWS Panorama Appliance supports H.264 video streams from cameras that output RTSP over a local network. The following camera models have been tested for compatibility with the AWS Panorama Appliance:

- Anpviz – IPC-B850W-S-3X, IPC-D250W-S
- Axis – M3057-PLVE, M3058-PLVE, P1448-LE, P3225-LV Mk II
- LaView – LV-PB3040W
- Vivotek – IB9360-H
- Amcrest – IP2M-841B
- **WGCC** – Dome PoE 4MP ONVIF

For the appliance's hardware specifications, see AWS Panorama Appliance Developer Kit specifications (p. 21).

AWS Panorama Appliance Developer Kit specifications

The AWS Panorama Appliance Developer Kit is a pre-production device for developing and testing AWS Panorama applications. It has the following hardware specifications.

> **Important**
> The AWS Panorama Appliance Developer Kit is not secured for use on production networks or workloads. For more information, see Security considerations for the AWS Panorama Appliance Developer Kit (p. 64).

Component	Specification
Processor and GPU	Nvidia Jetson Xavier AGX with 32GB RAM
Ethernet	2x 1000 Base-T (Gigabyte)
Wi-Fi	2x MIMO 802.11ac
USB	1x USB 2.0 and 1x USB 3.0 type-A female
HDMI output	2.0a
Dimensions	7.75” x 9.6” x 1.6” (197mm x 243mm x 40mm)
Weight	3.7lbs (1.7kg)
Power supply	100V-240V 50-60Hz AC 65W
Power input	IEC 60320 C6 (3-pin) receptacle
Dust and liquid protection	IP-62
EMI/EMC regulatory compliance	FCC Part-15 (US)
Thermal touch limits	IEC-62368
Operating temperature	-20°C to 60°C
Operating humidity	0% to 95% RH
Storage temperature	-20°C to 85°C
Storage humidity	Uncontrolled for low temperature. 90% RH at high temperature
Cooling	Forced-air heat extraction (fan)
Mounting options	Rackmount or free standing
Power cord	6-foot (1.8 meter)
Power control	Push-button
Reset	Momentary switch
Status and network LEDs	Programmable 3-color RGB LED

Bluetooth and SD card storage are present on the appliance but are not usable.

The AWS Panorama Appliance Developer Kit includes two screws for mounting on a server rack. You can mount two appliances side-by-side on a 19-inch rack.

AWS Panorama permissions

You can use AWS Identity and Access Management (IAM) to manage access to the AWS Panorama service and resources like appliances and applications. For users in your account that use AWS Panorama, you manage permissions in a permissions policy that you can apply to IAM users, groups, or roles. To manage permissions for the AWS Panorama Appliance, you add policies to a role that is specific to the appliance.

To manage permissions for users (p. 24) in your account, use the managed policy that AWS Panorama provides, or write your own. The AWS Panorama console uses multiple services to get information about your restrictive policies.

An AWS Panorama Appliance also has a role that grants it permission to access AWS services and resources. If your application accesses services with the AWS SDK, you grant it permission to call them in the appliance's role. The appliance's role is one of several service roles (p. 25) that the AWS Panorama service uses to access other services on your behalf.

You can restrict user permissions by the resource an action affects and, in some cases, by additional conditions. For example, you can specify a pattern for the Amazon Resource Name (ARN) of an application that requires a user to include their user name in the name of applications that they create. For the resources and conditions that are supported by each action, see Actions, resources, and condition keys for AWS Panorama in the Service Authorization Reference.

For more information, see What is IAM? in the IAM User Guide.

Topics

Identity-based IAM policies for AWS Panorama

To grant users in your account access to AWS Panorama, you use identity-based policies in AWS Identity and Access Management (IAM). Identity-based policies can apply directly to IAM users, or to IAM groups and roles that are associated with a user. You can also grant users in another account permission to assume a role in your account and access your AWS Panorama resources.

AWS Panorama provides managed policies that grant access to AWS Panorama API actions and, in some cases, access to other services used to develop and manage AWS Panorama resources. AWS Panorama updates the managed policies as needed, to ensure that your users have access to new features when they're released.

- **AWSPanoramaFullAccess** – Grants full access to AWS Panorama. View policy

Managed policies grant permission to API actions without restricting the resources that a user can modify. For finer-grained control, you can create your own policies that limit the scope of a user's permissions.

At a minimum, an AWS Panorama user needs permission to use the following services in addition to AWS Panorama:

- **Amazon Simple Storage Service (Amazon S3)** – To store model and Lambda function artifacts, and can be used for application output.
- **AWS Lambda** – To manage function code, configuration, and versions.
- **IAM** – To create a Lambda function, a user needs access to assign a role to the function. You can create an execution role with basic permissions ahead of time, in which case the user needs only permission to pass the role.

> **Creating service roles**
> The first time you use the AWS Panorama console, you need permission to create service roles (p. 25) used by the AWS Panorama service, the AWS Panorama console, the AWS Panorama Appliance, AWS IoT Greengrass, and SageMaker. A service role gives a service permission to manage resources or interact with other services. Create these roles before granting access to your users.

To create machine learning models or to monitor application output in the console, additional permissions are required. To use all features of AWS Panorama, also grant a user permission to use the following services:

- **Amazon SageMaker** – Develop, train, and compile machine learning models optimized for the AWS Panorama Appliance
- **Amazon CloudWatch** – View metrics output by AWS Panorama, Lambda, and other services
- **Amazon CloudWatch Logs** – View application logs

For details on the resources and conditions that you can use to limit the scope of a user's permissions in AWS Panorama, see Actions, resources, and condition keys for AWS Panorama in the Service Authorization Reference.

AWS Panorama service roles and cross-service resources

AWS Panorama uses other AWS services to manage the AWS Panorama Appliance, store data, and import application resources. These services also call other services using permissions managed with AWS Identity and Access Management (IAM) service roles. A service role gives a service permission to manage resources or interact with other services. When you sign in to the AWS Panorama console for the first time, you create the following service roles:

- **AWSPanoramaServiceRole** – Allows AWS Panorama to manage resources in Amazon Simple Storage Service (Amazon S3), AWS IoT, AWS IoT Greengrass, AWS Lambda, Amazon SageMaker, and Amazon CloudWatch, and to pass service roles to AWS IoT, AWS IoT Greengrass, and SageMaker

  Managed policy: AWSPanoramaServiceRolePolicy
- **AWSPanoramaApplianceRole** – Allows AWS IoT software on an AWS Panorama Appliance to upload logs to CloudWatch

  Managed policy: AWSPanoramaApplianceRolePolicy
- **AWSPanoramaSageMakerRole** – Allows SageMaker to manage objects in buckets created for use with AWS Panorama

  Managed policy: AWSPanoramaSageMakerRolePolicy
- **AWSPanoramaGreengrassGroupRole** – Allows a Lambda function on an AWS Panorama Appliance to manage resources in AWS Panorama, upload logs and metrics to CloudWatch, and manage objects in buckets created for use with AWS Panorama

  Managed policy: AWSPanoramaGreengrassGroupRolePolicy
- **AWSPanoramaGreengrassRole** – Grants AWS IoT Greengrass permission to access resources in Lambda, SageMaker, Amazon S3, and AWS IoT

  Managed policy: AWSGreengrassResourceAccessRolePolicy

To view the permissions attached to each role, use the IAM console. Wherever possible, the role's permissions are restricted to resources that match a naming pattern that AWS Panorama uses. For example, **AWSPanoramaServiceRole** grants only permission for the service to access Amazon S3 buckets that include the phrase `aws-panorama`.

AWS Panorama creates or accesses resources in the following services:

- AWS IoT – Things, policies, certificates, and jobs for the AWS Panorama Appliance
- Amazon S3 – Buckets and objects for model storage and application output
- AWS IoT Greengrass – A Greengrass resource that manages the AWS Panorama Appliance
- Lambda – Lambda functions that perform inference or process data in an application
- SageMaker – A training job that trains a model for use with an application
- CloudWatch Logs – Log groups and log streams for application logs
- IAM – Service roles and Lambda function runtime roles

For information about Amazon Resource Name (ARN) format or permission scopes for each service, see the topics in the *IAM User Guide* that are linked to in this list.

Managing the AWS Panorama Appliance

The AWS Panorama Appliance is the hardware that runs your applications. You use the AWS Panorama console to register an appliance, update its software, and deploy applications to it. The software on the AWS Panorama Appliance discovers and connects to camera streams, sends frames of video to your application, and displays video output on an attached display.

During the setup process, you connect cameras to the appliance for use with applications. You manage an appliance's cameras (p. 28) in the AWS Panorama console. When you deploy an application, you choose which camera streams the appliance sends to it for processing.

The AWS Panorama Appliance Developer Kit (p. 31) is a version of the AWS Panorama Appliance with developer features activated. You can connect to the AWS Panorama Appliance Developer Kit over a local network to run commands, view logs, and explore the AWS Panorama SDK.

For tutorials that introduce the AWS Panorama Appliance with a sample application, see Getting started with AWS Panorama (p. 3).

Topics

Managing an AWS Panorama Appliance

You use the AWS Panorama console to configure, upgrade or deregister the AWS Panorama Appliance.

To set up an appliance, follow the instructions in the getting started tutorial (p. 5). The setup process creates the resources in AWS Panorama that track your appliance and coordinate updates and deployments.

Sections

Update the appliance software

You view and deploy software updates for the AWS Panorama Appliance in the AWS Panorama console. Updates can be required or optional. When a required update is available, the console prompts you to apply it. You can apply optional updates on the appliance **Settings** page.

> **Important**
> When you update the appliance software, all data on the device is overwritten except for the contents of the `/data` directory. This directory includes your application data and logs. You can also use `/data` to store scripts and other files. For more information, see The AWS Panorama Appliance Developer Kit (p. 31).

To update the appliance software

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose **Settings**
4. Under **System software**, choose **Install version**.

Deregister an appliance

If you are done working with the AWS Panorama Appliance, you can use the AWS Panorama console to deregister it and delete the associated AWS IoT and AWS IoT Greengrass resources.

When you delete an appliance from the AWS Panorama service, data on the appliance is not deleted automatically. This data includes applications, camera information, the appliance certificate, network configuration, and logs. You can remove applications (p. 30) and cameras (p. 28) from the device prior to deregistering it, or reset the device to its factory state.

To delete an appliance

1. Open the AWS Panorama console Appliances page.
2. Choose the appliance.
3. Choose **Delete**.
4. Enter the appliance's name and choose **Delete**.

To fully reset the device and delete all data, press both the power button and the reset button for over 5 seconds. For more information, see AWS Panorama Appliance buttons and lights (p. 33).

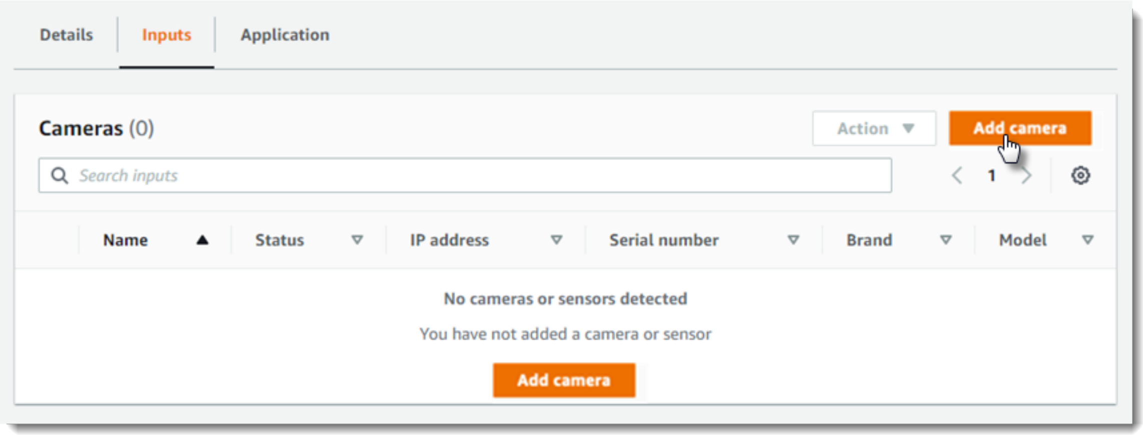
Managing camera streams for an AWS Panorama Appliance

To register video streams as data sources for your application, use the AWS Panorama console. An application can process multiple streams simultaneously and multiple appliances can connect to the same stream. The AWS Panorama Appliance can find camera streams automatically, or you can manually specify them.

> **Important**
> The AWS Panorama Appliance can connect to any camera stream that is routable from the local network it connects to. To secure your video streams, configure your network to allow only RTSP traffic locally. For more information, see Security in AWS Panorama (p. 52).

To add a camera stream to the AWS Panorama Appliance

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose **Inputs**.
4. Choose **Add camera**.



Cameras (0)

Name	Status	IP address	Serial number	Brand	Model

No cameras or sensors detected

You have not added a camera or sensor

5. Choose a connection mode. Try **Automatic** first. If it doesn't find your camera stream, use **Manual**.

   - **Automatic** – The AWS Panorama Appliance discovers cameras on the local network. Choose a camera and then choose a stream to add. If the camera has multiple streams, repeat the process to add additional streams.
   - **Manual** – Enter the IP address of the camera and the RTSP URL of a stream.

   Both workflows support password-protected cameras.
6. Choose **Confirm**.

For a list of cameras that are compatible with the AWS Panorama Appliance, see Supported computer vision models and cameras (p. 20).

Removing a stream

To deregister a video stream, remove it from the appliance's inputs.

To remove a camera stream

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose **Camera streams**.
4. Choose a stream.
5. Choose **Remove stream**.

Manage applications on an AWS Panorama Appliance

An application is a combination of Lambda functions, models, and configuration. From the **Appliances** page in the AWS Panorama console, you can modify an application's configuration and deploy published versions to the AWS Panorama Appliance.

To manage applications on an AWS Panorama Appliance

1. Open the AWS Panorama console Appliances page.
2. Choose an appliance.
3. Choose the **Applications** tab.

The **Applications** page shows applications that have been deployed to the appliance.

Use the options on this page to remove deployed applications from the appliance. To monitor deployments, choose **Application operations**.

The AWS Panorama Appliance Developer Kit

The AWS Panorama Appliance Developer Kit is a version of the AWS Panorama Appliance hardware that has developer features enabled. With the AWS Panorama Appliance Developer Kit, you can connect to the appliance over SSH to run commands.

> **Important**
> The AWS Panorama Appliance Developer Kit is not secured for use on production networks or workloads. For more information, see Security considerations for the AWS Panorama Appliance Developer Kit (p. 64).

If you are using the AWS Panorama Appliance Developer Kit for the first time, follow the tutorial at Using the AWS Panorama Appliance Developer Kit (p. 16). The tutorial provides an introduction to connecting to the device with SSH, viewing logs, and using Python libraries.

Sections

Connecting with SSH

If you configured SSH access when you set up the appliance (p. 5), you can connect to it from the command line to view logs, test Python scripts, and inspect the device's configuration.

To connect to the appliance with SSH, use the `ssh` command with the username and password that you configured during setup.

```
$ ssh 10.24.34.0 -l me
me@10.24.34.0's password:
```

Local storage

To store your application code locally on the appliance, use the `/data` directory. You can use this space to store scripts and other resources.

> **Warning**
> Do not use the user directory under `/home` for local storage. This directory is reset when you update your appliance. Files that you add there are deleted.

Granting additional permissions

The AWS Panorama Appliance Developer Kit has an AWS Identity and Access Management (IAM) role that grants it limited permission to access resources in your account. You create this role, AWSPanoramaGreengrassGroupRole, when you first use the AWS Panorama console.

By default, the appliance has permission to upload logs to CloudWatch Logs, to post metrics to CloudWatch, and to access objects in buckets that you create for use with AWS Panorama. You can add permissions to the role to give the appliance and your application code permission to use additional AWS services.

To extend the appliance's permissions

1. Open the AWSPanoramaGreengrassGroupRole role in the IAM console.

2. Choose **Attach policies**.
3. Attach a policy that grants permission to use an AWS service.

For more information on permissions in AWS Panorama, see AWS Panorama permissions (p. 23).

Logs

After connecting to the appliance with SSH, you can find logs in the following locations.

- **Application code** – `/data/greengrass/ggc/var/log/user/`*`us-east-1`*`/`*`123456789012`*`/`*`function-name`*`.log`

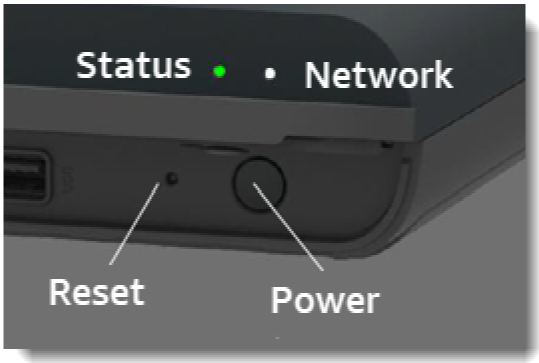
  Replace the highlighted values with your AWS Region, account ID, and function name.
- **Appliance software** – `/var/log/syslog`, `/opt/aws/panorama/mediapipeline`
- **AWS IoT Greengrass system** – `/data/greengrass/ggc/var/log/system`
- **AWS IoT agents** – `/opt/aws/panorama/iot`
- **CloudWatch Logs agent** – `/var/log/awslogs.log`, `/var/log/log_daemon.log`

Most of these logs are also sent to CloudWatch Logs. For more information, see Monitoring AWS Panorama resources and applications (p. 46).

For a tutorial that introduces the AWS Panorama Appliance Developer Kit and debugging, see Using the AWS Panorama Appliance Developer Kit (p. 16)

AWS Panorama Appliance buttons and lights

The AWS Panorama Appliance has two LED lights above the power button that indicate the device status and network connectivity.



The LEDs change color and blink to indicate status. A slow blink is once every three seconds. A fast blink is once per second.

Status LED states

- **Fast blinking green** – The appliance is booting up.
- **Solid green** – The appliance is operating normally.
- **Slow blinking blue** – The appliance is copying configuration files and attempting to register with AWS IoT Greengrass.
- **Fast blinking red** – The appliance encountered an error during startup or is overheated.
- **Slow blinking orange** – The appliance is restoring the latest software image.
- **Fast blinking orange** – The appliance is restoring the factory software image.

To reset the device, use the following button combinations. A short press is 1 second. A long press is 5 seconds. For operations that require multiple buttons, press and hold both buttons simultaneously.

Reset operations

- **To shut down the appliance** – Short press power. Starts a shutdown sequence that takes about 10 seconds.
- **To restore the latest software image** – Shut down the appliance, and then short press reset.
- **To restore the factory software image** – Shut down the appliance, and then long press reset.
- **To restore the factory software image and delete all configuration files and applications** – Shut down the appliance, and then long press power and reset.

The network LED has the following states:

Network LED states

- **Solid green** – An Ethernet cable is connected.
- **Blinking green** – The appliance is communicating over the network.
- **Solid red** – An Ethernet cable is not connected.
- **Solid blue** – An Ethernet cable is not connected, but Wi-Fi is enabled.

Building AWS Panorama applications

Applications run on the AWS Panorama Appliance to perform computer vision tasks on video streams. You can build computer vision applications by combining Python code and machine learning models, and deploy them to the AWS Panorama Appliance over the internet. Applications can send video to a display, or use the AWS SDK to send results to AWS services.

A model (p. 37) analyzes images to detect people, vehicles, and other objects. Based on images that it has seen during training, the model tells you what it thinks something is, and how confident it is in its guess. You can train models with your own image data or get started with a sample.

The application's code (p. 40) loads a model, sends still images to it, and processes the result. A model might detect multiple objects and return their shapes and location. The code can use this information to add text or graphics to the video, or to send results to an AWS service for storage or further processing.

To get images from a stream, interact with a model, and output video, application code uses the AWS Panorama Application SDK (p. 45). The application SDK is a Python library that supports models generated with PyTorch, Apache MXNet, and TensorFlow.

Topics

Managing applications and application versions in AWS Panorama

Use the AWS Panorama console to manage applications and application versions. An *application* is a computer vision program that runs on the AWS Panorama Appliance. *Application versions* are immutable snapshots of an application's configuration. AWS Panorama saves previous versions of your applications so that you can roll back updates that aren't successful, or run different versions on different appliances.

To create an application, you need an AWS Lambda function and a computer vision model that is stored in Amazon SageMaker or Amazon Simple Storage Service (Amazon S3). The application code is a Python script that uses the AWS Panorama Application SDK to process inputs, run inference, and output video. If you have not created an application yet, see Getting started with AWS Panorama (p. 3) for a walkthrough.

Sections

Deploy an application

To deploy an application, use the AWS Panorama console. During the deployment process, you choose which camera streams to pass to the application code, and whether to send the output to a display.

To deploy an application

1. Open the AWS Panorama console Applications page.
2. Choose an application.
3. (Optional) To deploy a previous version of the application, for **Version**, choose the version you want to deploy.
4. Choose **Deploy**.
5. Follow the instructions to deploy the application version.

The deployment process takes a few minutes. The appliance's output can be blank for an extended period while the application starts. If you encounter an error, see Troubleshooting (p. 49).

Update or copy an application

To update an application or create a copy of it, use the **Clone** option. AWS Panorama saves all versions of applications until you delete them. When you clone an application, you can update its function or models.

To clone an application

1. Open the AWS Panorama console Applications page.
2. Choose an application.
3. Choose **Clone**.
4. Follow the instructions to create a new version or application.

Delete versions and applications

To clean up unused application versions, delete them. When you delete an application version, AWS Panorama removes it from connected appliances. If an appliance is offline when its application is deleted, you can remove it from the appliance (p. 30) after it reconnects.

To delete an application version

1. Open the AWS Panorama console Applications page.
2. Choose an application.
3. For **Version**, choose a version.
4. Choose **Delete**.

An application must have at least one version. To remove all versions, delete the application.

To delete an application

1. Open the AWS Panorama console Applications page.
2. Choose an application.
3. Choose **Delete application**.

Computer vision models

A *computer vision model* is a software program that is trained to detect objects in images. A model learns to recognize a set of objects by first analyzing images of those objects through training. A computer vision model takes an image as input and outputs information about the objects that it detects, such as the type of object and its location. AWS Panorama supports computer vision models built with PyTorch, Apache MXNet, and TensorFlow.

> **Note**
> For a list of pre-built models that have been tested with AWS Panorama, see Model compatibility.

You can use a sample model (p. 37) or build your own. A model can detect multiple objects in an image, and each result can have multiple outputs, such as the name of a class, a confidence rating, and a bounding box. You can train a model outside of AWS and store it in Amazon Simple Storage Service (Amazon S3), or train it with Amazon SageMaker. To build a model in SageMaker, you can use the built-in image classification algorithm. AWS Panorama can reference the training job to find the trained model that it created in Amazon S3.

> **Important**
> Whether you import a model from SageMaker or from Amazon S3, the name of the Amazon S3 bucket where the model is stored must contain `aws-panorama`. The service role (p. 25) that gives AWS Panorama permission to access objects in Amazon S3 enforces this naming requirement.

Sections

Sample model

This guide uses a sample object detection model. The sample model uses the object detection algorithm to identify multiple objects in an image. For each object, the model outputs the type of object, a confidence score, and coordinates of a bounding box. It uses the Single Shot multibox detector (SSD) framework and the ResNet base network.

- Download the sample model

To get started with the sample model, see Deploying the AWS Panorama sample application (p. 8).

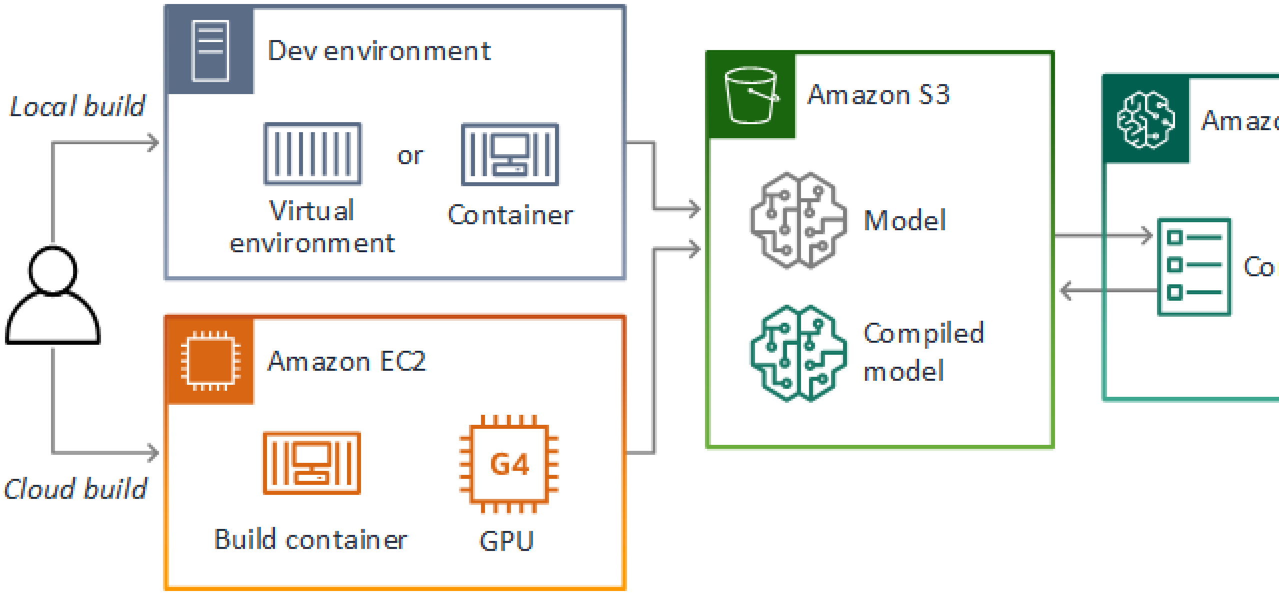
Building a custom model

You can use models that you build in PyTorch, Apache MXNet, and TensorFlow in AWS Panorama applications. As an alternative to building and training models in SageMaker, you can use a trained model or build and train your own model with a supported framework and export it in a local environment or in Amazon EC2.

> **Note**
> For details about the framework versions and file formats supported by SageMaker Neo, see Supported Frameworks in the Amazon SageMaker Developer Guide.

The repository for this guide provides a sample application that demonstrates this workflow for a Keras model in TensorFlow `SavedModel` format. It uses TensorFlow 1.15 and can run locally in a virtual environment or in a Docker container. The sample app also includes templates and scripts for building the model on an Amazon EC2 instance.

- Custom model sample application



AWS Panorama uses SageMaker Neo to compile models for use on the AWS Panorama Appliance. For each framework, use the format that's supported by SageMaker Neo, package the model in a `.tar.gz` archive, and store it in an Amazon S3 bucket that AWS Panorama can access.

For more information, see Compile and Deploy Models with Neo in the Amazon SageMaker Developer Guide.

Using models in code

On the appliance, model files are stored in a folder named after the model resource that you create in the AWS Panorama console when you create an application (p. 9). The application code uses the directory name to reference the model and load it with the AWS Panorama Application SDK.

For example, the following initialization code loads a model named `my-model`.

Example lambda_function.py – Initialization

```
def init(self, parameters, inputs, outputs):
    try:
        self.threshold = parameters.threshold
        self.person_index = parameters.person_index
        self.frame_num = 0
        self.number_people = 0
        self.colours = np.random.rand(32, 3)

        self.model = panoramasdk.model()
        self.model.open('my-model', 1)
```

```
            print("Creating input and output arrays")
            class_info = self.model.get_output(0)
            prob_info = self.model.get_output(1)
            rect_info = self.model.get_output(2)

            self.class_array = np.empty(class_info.get_dims(), dtype=class_info.get_type())
            self.prob_array = np.empty(prob_info.get_dims(), dtype=prob_info.get_type())
            self.rect_array = np.empty(rect_info.get_dims(), dtype=rect_info.get_type())
```

Training models

When you train a model, use images from the target environment, or from a test environment that closely resembles the target environment. Consider the following factors that can affect model performance:

- **Lighting** – The amount of light that is reflected by a subject determines how much detail the model has to analyze. A model trained with images of well-lit subjects might not work well in a low-light or backlit environment.
- **Resolution** – The input size of a model is typically fixed at a resolution between 224 and 512 pixels wide in a square aspect ratio. Before you pass a frame of video to the model, you can downscale or crop it to fit the required size.
- **Image distortion** – A camera's focal length and lens shape can cause images to exhibit distortion away from the center of the frame. The position of a camera also determines which features of a subject are visible. For example, an overhead camera with a wide angle lens will show the top of a subject when it's in the center of the frame, and a skewed view of the subject's side as it moves farther away from center.

To address these issues, you can preprocess images before sending them to the model, and train the model on a wider variety of images that reflect variances in real-world environments. If a model needs to operate in a lighting situations and with a variety of cameras, you need more data for training. In addition to gathering more images, you can get more training data by creating variations of your existing images that are skewed or have different lighting.

Authoring application code

In your application code, you use the AWS Panorama Application SDK to load the model, run inference, and output results. The AWS Panorama Application SDK is a Python library that is available in the runtime environment on the AWS Panorama Appliance. The SDK defines a base class for applications and provides methods for loading a model and using it to process images.

AWS Panorama use AWS Lambda and AWS IoT Greengrass to manage and deploy application code. In Lambda, you create a Python 3.7 function (p. 8) and publish a version that stores your application's code. When you deploy an application, AWS Panorama uses AWS IoT Greengrass to deploy the code and the application's model to the AWS Panorama Appliance.

In your application code, you can also use the AWS SDK for Python (Boto3) and common machine learning and computer vision libraries like NumPy and OpenCV. With the SDK for Python (Boto3), you can use AWS services to store results or perform additional processing. For more information, see Calling AWS services from your application code (p. 43).

Sections

Using the base class

To create an application class, declare `panoramasdk.base` as the parent class and define the following methods:

- `interface` – Returns a dictionary that defines the class's parameters, input type, and output type
- `init` – Creates resources that are reused, such as the application's model and AWS SDK clients
- `entry` – Processes a frame of video from one or more camera streams
- `main` – Instantiates the class and calls `run()` on it

The following code shows the basic outline of an application class.

Example my_application.py – Class methods

```
import panoramasdk

class my_application(panoramasdk.base):

  def interface(self):
    ...

  def init(self, parameters, inputs, outputs):
    ...

  def entry(self, inputs, outputs):
    ...

def main():
  my_application().run()
```

```
main()
```

The `run` method is part of the parent class. It tells the AWS Panorama Application SDK that the application is ready to process video streams.

Defining inputs and outputs

The `interface` method returns an object that defines the application's parameters, inputs, and outputs.

```
  def interface(self):
    return {
      "parameters":
        (
          ("model", "model_name", "Name of the model in AWS Panorama", "aws-panorama-
sample-model"),
        ),
      "inputs":
        (
          ("media[]", "video_in", "Camera input stream"),
        ),
      "outputs":
        (
          ("media[video_in]", "video_out", "Camera output stream"),
        )
    }
```

The `media` array under `inputs` represents input from cameras connected to the appliance. The application SDK fills this array with frames of video (one for each stream) and passes it to the entry method (p. 42). Similarly, the `outputs` object provides a place to write an output image after processing the input.

The `parameters` object defines the application's runtime configuration. In the preceding example, the `model_name` parameter has type `model` and its value is `aws-panorama-sample-model`. The value of `model_name` is accessible by referencing `parameters.model_name`. To avoid hard-coding names and numbers in your application logic, define them in the interface object.

Parameters can have the following types.

- `media` – A frame of video
- `model` – A computer vision model
- `rect` – A rectangle of (x_0, y_0, x_1, y_1)
- `string` – A string
- `uint32` – A 4-byte, unsigned integer
- `int` – An integer
- `float` – A floating-point value
- `byte` – A byte
- `bool` – `True` or `False`

Loading a model

In the `init` method, load the application's model and create other reusable resources, such as AWS SDK clients.

```
  def init(self, parameters, inputs, outputs):
```

```
  try:
    print("Loading model: " + parameters.model_name)
    self.model = panoramasdk.model()
    self.model.open(parameters.model_name, 1)
    return True
  except Exception as e:
    print("Exception: {}".format(e))
    return False
```

Processing frames

The `entry` method processes an array of frames from one or more video streams. The input is an array of `media` objects that represent frames of video. If the application is connected to one camera stream, `inputs.video_in` has only one object.

The following example shows an application getting a frame from each attached camera stream, calling a local `preprocess` method to resize and normalize it, and processing the result with a model.

```
def entry(self, inputs, outputs):

  for i in range(len(inputs.video_in)):
    frame = inputs.video_in[i]
    frame.add_label('AWS Panorama', 0.8, 0.05)
    frame_image = frame.image

    # Prepare the image and run inference
    normalized_image = self.preprocess(frame_image)
    self.model.batch(0, normalized_image)
    self.model.flush()
    resultBatchSet = self.model.get_result()
    ...

    self.model.release_result(resultBatchSet)
    outputs.video_out[i] = frame

  return True
```

Calling AWS services from your application code

You can use the AWS SDK for Python (Boto) to call AWS services from your application code. For example, if your model detects something out of the ordinary, you could post metrics to Amazon CloudWatch, send an notification with Amazon SNS, save an image to Amazon S3, or invoke a Lambda function for further processing. Most AWS services have a public API that you can use with the AWS SDK.

The appliance does not have permission to access all AWS services by default. To grant it permission, add the API actions that it uses to the appliance's role (AWSPanoramaGreengrassGroupRole (p. 25)). You create this role when you first use the AWS Panorama console. It comes with limited permission to use Amazon S3 CloudWatch, and Amazon CloudWatch Logs.

Sections

Using Amazon S3

You can use Amazon S3 to store processing results and other application data.

```
import boto3
s3_client=boto3.client("s3")
s3_clients3.upload_file(data_file,
                    s3_bucket_name,
                    os.path.basename(data_file))
```

The appliance has permission to access buckets that include `aws-panorama` in the name. To grant it additional permission, you can modify the appliance's role (AWSPanoramaGreengrassGroupRole (p. 25)).

Using the AWS IoT MQTT topic

You can use the SDK for Python (Boto3) to send messages to an MQTT topic in AWS IoT. In the following example, the application posts to a topic named after the appliance's *thing name*, which you can find in AWS IoT console.

```
import boto3
iot_client=boto3.client('iot-data')
topic = "panorama/panorama_my-appliance_Thing_a01e373b"
iot_client.publish(topic=topic, payload="my message")
```

The topic name can be anything. To publish messages, the device needs permission to call `iot:Publish`.

To monitor an MQTT queue

1. Open the AWS IoT console Test page.
2. For **Subscription topic**, enter the name of the topic. For example, `panorama/panorama_my-appliance_Thing_a01e373b`.
3. Choose **Subscribe to topic**.

Adding text and boxes to output video

With the AWS Panorama SDK, you can output a video stream to a display. The video can include text and boxes that show output from the model, the current state of the application, or other data.

Each object in the `video_in` array is an image from a camera stream that is connected to the appliance. The type of this object is `panoramasdk.media`. It has methods to add text and rectangular boxes to the image, which you can then assign to the `video_out` array.

In the following example, the sample application overlays the number of people in frame, and draws a box around each. It gets the coordinates of the box from the model, which outputs the location of each result, in addition to the class and confidence score. It uses the `add_rect` method to draw the bounding boxes, and the `add_label` method to add text.

Example lambda_function.py – Bounding boxes and text

```
def entry(self, inputs, outputs):
    for i in range(len(inputs.video_in)):
        stream = inputs.video_in[i]
        person_image = stream.image
        ...
        # Draw bounding boxes on output image
        if self.number_people > 0:
            for index in person_indices:

                left = np.clip(rect_data[index][0] / np.float(HEIGHT), 0, 1)
                top = np.clip(rect_data[index][1] / np.float(WIDTH), 0, 1)
                right = np.clip(rect_data[index][2] / np.float(HEIGHT), 0, 1)
                bottom = np.clip(rect_data[index][3] / np.float(WIDTH), 0, 1)

                stream.add_rect(left, top, right, bottom)
                stream.add_label(str(prob_data[index][0]), right, bottom)
        # Add text
        stream.add_label('Number of People : {}'.format(self.number_people), 0.8, 0.05)
        ...
        outputs.video_out[i] = stream
```

In this example, (`x_coordinate=0.1, y_coordinate=0.1`) means the top-left corner of the text is placed 10% of the width and height away from the top-left corner of the image frame.

For more information about the AWS Panorama application SDK, see The AWS Panorama Application SDK (p. 45).

The AWS Panorama Application SDK

The AWS Panorama Application SDK is a Python library for developing AWS Panorama applications. In your application code (p. 40), you use the AWS Panorama Application SDK to load a computer vision model, run inference, and output video to a monitor.

> **Note**
> To ensure that you have access to the latest functionality of the AWS Panorama Application SDK, upgrade the appliance software (p. 27).

To interact with the AWS Panorama Application SDK., you can connect to the AWS Panorama Appliance Developer Kit with SSH. With a Python interpreter, you can load the AWS Panorama Application SDK and verify its behavior.

```
$ python3
Python 3.7.5

>>> import panoramasdk
>>> help(panoramasdk)

    CLASSES
        builtins.list(builtins.object)
            port
        builtins.object
            base
            batch
            batch_set
    ...
```

For an introduction to using the AWS Panorama Appliance Developer Kit for development, see Using the AWS Panorama Appliance Developer Kit (p. 16).

For details about the classes that the application SDK defines and their methods, see Application SDK reference.

Monitoring AWS Panorama resources and applications

You can monitor AWS Panorama resources in the AWS Panorama console and with Amazon CloudWatch. The AWS Panorama Appliance connects to the AWS Cloud over the internet to report its status and the status of connected cameras. While it is on, the appliance also sends logs to CloudWatch Logs in real time.

The appliance gets permission to use AWS IoT, CloudWatch Logs, and other AWS services from a service role that you create the first time that you use the AWS Panorama console. For more information, see AWS Panorama service roles and cross-service resources (p. 25).

For help troubleshooting specific errors, see Troubleshooting (p. 49).

Topics

Monitoring in the AWS Panorama console

You can use the AWS Panorama console to monitor your AWS Panorama Appliance and cameras. The console uses AWS IoT to monitor the state of the appliance.

To monitor your appliance and cameras in the AWS Panorama console

1. Open the AWS Panorama console.
2. Open the AWS Panorama console Appliances page.
3. Choose an appliance.
4. To see the status of the appliance's network interfaces, choose **Settings**.
5. To see the status of connected cameras, choose **Camera streams**.

The overall status of the appliance appears at the top of the page. If the status is **Online**, then the appliance is connected to AWS and sending regular status updates.

Viewing AWS Panorama event logs in CloudWatch Logs

AWS Panorama reports application and system events to Amazon CloudWatch Logs. When you encounter issues, you can use the event logs to help debug your AWS Panorama application or to troubleshoot the application configuration.

To debug your code and validate that it is working as expected, you can output logs with the standard logging functionality for Python. You can view logs in the CloudWatch Logs console.

To view logs in CloudWatch Logs

1. Open the Log groups page of the CloudWatch Logs console.
2. Find AWS Panorama application and appliance logs in the following groups:

   - **Application code** – `/aws/greengrass/Lambda/us-east-1/123456789012/function-name`

     Replace the highlighted values with your AWS Region, account ID, and function name.
   - **AWS IoT Greengrass system** – `/aws/greengrass/GreengrassSystem/component-name`
   - **AWS Panorama Appliance system** – `/aws/panorama_device/iot-thing-name`

     Log streams include `syslog`, `iot_job_agent`, `mediapipeline`, and a stream for each camera.

If you don't see logs in CloudWatch Logs, confirm that you are in the correct AWS Region. If you are, there might be an issue with the appliance's connection to AWS or with permissions on the appliance's AWS Identity and Access Management (IAM) role (p. 25).

With the AWS Panorama Appliance Developer Kit, you can connect to the appliance over a local network to view logs and configuration files. For more information, see Using the AWS Panorama Appliance Developer Kit (p. 16).

Troubleshooting

The following topics provide troubleshooting advice for errors and issues that you might encounter when using the AWS Panorama console, appliance, or SDK. If you find an issue that is not listed here, use the **Provide feedback** button on this page to report it.

You can find logs for your appliance in the Amazon CloudWatch Logs console. The appliance uploads logs from your application code, the appliance software, and AWS IoT processes as they are generated. For more information, see Viewing AWS Panorama event logs in CloudWatch Logs (p. 48).

For more troubleshooting advice and answers to common support questions, visit the AWS Knowledge Center.

Application configuration

Error: *Unable to parse model configuration ... parseFile(1145) unable to open file*

The name of the model in AWS Panorama must match the name of the model parameter in your application code. You refer to this name when you load a model with the `panoramasdk` module.

Error: *Failed to initialize Lambda runtime due to exception: initialize(108) Node enable signal is not set, terminating*

The software that runs your application code waits for a signal from another process before loading the Lambda function handler. If your code returns an error, the application node is deactivated, causing this error to appear repeatedly in the application log. Check the log from when the application deployment completed to find the original error that caused the node to be deactivated. You can also search for the text *Failed to call Lambda entry method* which appears when your code throws an exception.

Appliance configuration

Error: *Resource temporarily unavailable* (SSH)

Check the network connection to the appliance. A VPN client or firewall software can block connections.

Issue: *The appliance shows a blank screen during boot up.*

After completing the initial boot sequence, which takes about one minute, the appliance shows a blank screen for a minute or more while it loads your model and starts your application. Also, the appliance does not output video if you connect a display after it turns on.

Issue: *The appliance doesn't respond when I hold the power button down to turn it off.*

The appliance takes up to 10 seconds to shut down safely. You need to hold the power button down for only 1 second to start the shutdown sequence. For a complete list of button operations, see AWS Panorama Appliance buttons and lights (p. 33).

Issue: *I need to generate a new configuration archive to change settings or replace a lost certificate.*

AWS Panorama does not store the device certificate or network configuration after you download it. If you still have the archive and need to modify the network configuration, you can extract the network.json file, modify it, and update the archive. If you don't have the archive, you can delete the appliance using the AWS Panorama console and create a new one with a new configuration archive.

Camera configuration

Error: *No camera has been found in the subnet. Please verify that there is at least one camera in the subnet*

Automatic camera discovery uses ONVIF profile S to search for camera streams on the appliance's local network. If your camera does not support ONVIF or if automatic discovery doesn't work for some other reason, you can connect to a stream manually by entering the camera's IP address and RTSP stream URL. For example: `rtsp://10.0.0.99/live/mpeg4`. The path part of the URL varies depending on your camera.

Error: *Failed to set up the "video/H264" subsession: 461 Unsupported transport*

Error: *Bit stream out of bounds.*

Error: *initNal(63) Error encountered while processing NAL.*

If you can connect the appliance to the camera in the AWS Panorama console, but errors such as these appear in the appliance logs, there may be an issue with the camera or the appliance software. Verify that your camera supports H.264 streams and is configured to use H.264 if multiple codecs are available.

Use the **Provide feedback** link on this page to send us the camera model and error. For a list of cameras that have been tested for compatibility with the AWS Panorama Appliance, see Supported cameras (p. 20).

Model compatibility

Error: *ClientError: OperatorNotImplemented:('Operator `Model` is not supported for frontend `Keras`.*

Your model's architecture is not supported. When you deploy a model, AWS Panorama uses SageMaker Neo to compile it to run on the AWS Panorama Appliance. SageMaker Neo supports a subset of layer types for each model framework.

For more information, see Supported models (p. 20).

Error: *ClientError: InputConfiguration: No valid TensorFlow model found in input files. Please make sure the framework you select is correct.*

For TensorFlow models, you must use the SavedModel format. If you create a TensorFlow model with Keras, call `model.save` with a directory name, not a filename that has a `.h5` or `.keras` extension. Including these extensions creates an H5-format model file, which is not supported.

Error: *ClientError: InputConfiguration: TVM cannot convert Tensorflow model. Please make sure the framework you selected is correct.*

This error can occur if you specify the input shape of the model incorrectly. To find the input shape, check the configuration of your input layer. For example, in Keras, you can use the `summary` method on the model, or `get_config` on the layer.

In the following examples, the input name is `input_1` and the input shape is `1,224,224,3`. The first value is the batch size, which is not hard-coded in the model. This is followed by the image width and height, and the number of color channels. In some models, the number of channels comes before the height and width.

Example Model summary

```
Model: "mobilenetv2_1.00_224"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to

==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
```

Example Layer configuration

```
{'batch_input_shape': (None, 224, 224, 3),
 'dtype': 'float32',
 'sparse': False,
 'ragged': False,
 'name': 'input_1'}
```

Security in AWS Panorama

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

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

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

Important
The AWS Panorama Appliance Developer Kit is not secured for use on production networks or workloads. For more information, see Security considerations for the AWS Panorama Appliance Developer Kit (p. 64).

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

Topics

Data protection in AWS Panorama

The AWS shared responsibility model applies to data protection in AWS Panorama. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.

- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with AWS Panorama or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Sections

Encryption in transit

AWS Panorama API endpoints support secure connections only over HTTPS. When you manage AWS Panorama resources with the AWS Management Console, AWS SDK, or the AWS Panorama API, all communication is encrypted with Transport Layer Security (TLS). Communication between the AWS Panorama Appliance and AWS is also encrypted with TLS. Communication between the AWS Panorama Appliance and cameras over RTSP is not encrypted.

For a complete list of API endpoints, see AWS Regions and endpoints in the *AWS General Reference*.

Encryption at rest

The AWS Panorama service does not copy or store data such as machine learning models and application code. These artifacts are stored in other services and AWS Panorama uses AWS IoT Greengrass to deploy them to the AWS Panorama Appliance Developer Kit. Configuration files, models, and code are not encrypted at rest on the AWS Panorama Appliance Developer Kit.

The contents of the configuration archive, which includes the appliance's private key and network configuration, are not encrypted. The network configuration file can contain Wi-Fi passwords and SSH credentials in plain text. AWS Panorama does not store these files; they can only be retrieved when you register an appliance. After you transfer the configuration archive to an appliance, delete it from your computer and USB storage device.

Other settings, such as camera stream credentials (username and password) are encrypted at rest in AWS. Settings are decrypted prior to transport and sent to the appliance over TLS.

To store your models securely in Amazon S3, you can use server-side encryption with a key that Amazon S3 manages, or one that you provide. For more information, see Protecting data using encryption in the Amazon Simple Storage Service User Guide.

When you author application code in AWS Lambda, Lambda encrypts the function code by default. For more information, see Data protection in AWS Lambda in the AWS Lambda Developer Guide.

The AWS Panorama Appliance sends log data to Amazon CloudWatch Logs. CloudWatch Logs encrypts this data by default, and can be configured to use a customer managed key. For more information, see Encrypt log data in CloudWatch Logs using AWS KMS in the Amazon CloudWatch Logs User Guide.

Identity and access management for AWS Panorama

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

Topics

Audience

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

Service user – If you use the AWS Panorama service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Panorama features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS Panorama, see Troubleshooting AWS Panorama identity and access (p. 60).

Service administrator – If you're in charge of AWS Panorama resources at your company, you probably have full access to AWS Panorama. It's your job to determine which AWS Panorama features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS Panorama, see How AWS Panorama works with IAM (p. 58).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Panorama. To view example AWS Panorama identity-based policies that you can use in IAM, see AWS Panorama identity-based policy examples (p. 58).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

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

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

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

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

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

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for AWS Panorama in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

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

Identity-based policies

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

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

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

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

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

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies.

Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

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

How AWS Panorama works with IAM

Before you use IAM to manage access to AWS Panorama, you should understand what IAM features are available to use with AWS Panorama. To get a high-level view of how AWS Panorama and other AWS services work with IAM, see AWS services that work with IAM in the *IAM User Guide*.

For an overview of permissions, policies, and roles as they are used by AWS Panorama, see AWS Panorama permissions (p. 23).

AWS Panorama identity-based policy examples

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

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

Topics

Policy best practices

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

- **Get started using AWS managed policies** – To start using AWS Panorama quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to

specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

Using the AWS Panorama console

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

For more information, see Identity-based IAM policies for AWS Panorama (p. 24)

Allow users to view their own permissions

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

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

AWS managed policies for AWS Panorama

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

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

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

AWS Panorama provides the following managed policies. For the full contents and change history of each policy, see the linked pages in the IAM console.

- AWSPanoramaFullAccess – Provides full access to AWS Panorama.
- AWSPanoramaApplianceRolePolicy – Allows AWS IoT software on an AWS Panorama Appliance to upload logs to Amazon CloudWatch.
- AWSPanoramaGreengrassGroupRolePolicy – Allows an AWS Lambda function on an AWS Panorama Appliance to manage resources in Panorama, upload logs and metrics to Amazon CloudWatch, and to manage objects in buckets created for use with Panorama.
- AWSPanoramaSageMakerRolePolicy – Allows Amazon SageMaker to manage objects in buckets created for use with AWS Panorama.
- AWSPanoramaServiceRolePolicy – Allows AWS Panorama to manage resources in Amazon S3, AWS IoT, AWS IoT GreenGrass, AWS Lambda, Amazon SageMaker, and Amazon CloudWatch Logs, and to pass service roles to AWS IoT, AWS IoT GreenGrass, and Amazon SageMaker.

Troubleshooting AWS Panorama identity and access

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

Topics

I am not authorized to perform an action in AWS Panorama

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a function but does not have `panorama:DescribeAppliance` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 panorama:DescribeAppliance on resource: my-appliance
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-appliance` resource using the `panorama:DescribeAppliance` action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to AWS Panorama.

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

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

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

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access AWS Panorama

To allow others to access AWS Panorama, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS Panorama.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

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

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

To learn more, consult the following:

- To learn whether AWS Panorama supports these features, see How AWS Panorama works with IAM (p. 58).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Compliance validation for AWS Panorama

AWS Panorama is not in scope of any AWS compliance programs. For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS compliance programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS artifact.

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

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

Additional considerations for when people are present

Below are some best practices to consider when using AWS Panorama for scenarios where people might be present:

- Ensure that you are aware of and compliant with all applicable laws and regulations for your use case. This may include laws related to the positioning and field of view of your cameras, notice and signage requirements when placing and using cameras, and the rights of people that may be present in your videos, including their privacy rights.

- Take into account the effect of your cameras on people and their privacy. In addition to legal requirements, consider whether it would be appropriate to place notice in areas where your cameras are located, and whether cameras should be placed in plain sight and free of any occlusions, so people are not surprised that they may be on camera.
- Have appropriate policies and procedures in place for the operation of your cameras and review of data obtained from the cameras.
- Consider appropriate access controls, usage limitations, and retention periods for the data obtained from your cameras.

Resilience in AWS Panorama

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

The list of available services varies by AWS Region. For more information about AWS Regions and Availability Zones, see AWS global infrastructure.

Infrastructure security in AWS Panorama

As a managed service, AWS Panorama is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of security processes whitepaper.

You use AWS published API calls to access AWS Panorama through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

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

AWS Panorama Appliance network activity

The AWS Panorama Appliance connects to your cameras over a local Ethernet connection and uses TLS on port 443 to connect to the AWS Cloud. The endpoint host name is not fixed, but the endpoint format is *.iot.<aws_region>.amazonaws.com. For more information see AWS IoT Core endpoints and quotas in the *AWS General Reference guide*.

AWS Panorama uses your network for the following AWS Panorama Appliance functions.

Application deployment

AWS Panorama uses your network to connect to AWS IoT Greengrass and uses remote AWS IoT jobs to deploy your application to your AWS Panorama Appliance. The AWS IoT Greengrass core manages the network activity of AWS IoT Greengrass. For more information see Configure the AWS IoT Greengrass core.

Camera identification and connection

When you add camera streams in Automatic connection mode, the AWS Panorama Appliance uses a remote AWS IoT job to scan your network and identify ONVIF compliant cameras on your subnet. The AWS Panorama Appliance then uses Real-time Streaming Protocol (RTSP) to connect to the video streams from IP cameras on your network. For initial setup, you must use RTSTP on port 554 for streaming. After initial setup, RTSP can use additional ports.

For more information see the RTSP RFC 2326.

Continuous monitoring

AWS Panorama uses a remote AWS IoT job to monitor the network status and software version on your AWS Panorama Appliance. The job runs every 30 seconds to determine whether your AWS Panorama Appliance is online and using the latest software.

AWS Panorama Appliance updates

When you request a software update or provide new credentials or configurations, AWS Panorama uses a remote AWS IoT job to send the updates to your AWS Panorama Appliance over the network.

Security considerations for the AWS Panorama Appliance Developer Kit

The AWS Panorama Appliance Developer Kit helps you develop a proof of concept for your computer vision applications. We recommend using the AWS Panorama Appliance Developer Kit only in a test environment. We don't recommend simultaneously connecting the developer kit to a sensitive or isolated network and a network with production camera streams. Be careful if your configuration allows the AWS Panorama Appliance Developer Kit to act as a bridge to a sensitive IP camera network.

You are responsible for the following:

- The physical and logical network security of the AWS Panorama Appliance Developer Kit.
- Securely operating the network-attached cameras when you use the AWS Panorama Appliance Developer Kit.
- Keeping the AWS Panorama Appliance and camera software updated.
- Complying with any applicable laws or regulations associated with the content of the videos and images you gather from your production environments, including those related to privacy.

The AWS Panorama Appliance Developer Kit uses unencrypted RTSP camera streams. For details on network activity and ports, see AWS Panorama Appliance network activity (p. 63). For details on encryption, see Data protection in AWS Panorama (p. 52).

Network configuration for testing with non-production data (most secure)

For testing with non-production data with the AWS Panorama Appliance Developer Kit, we recommend that you use a local IP camera and physically connect the camera to AWS Panorama Appliance using an Ethernet connection. Then use a subnet (isolated from your production environment) to connect the AWS Panorama Appliance to the AWS Cloud.

If a physical Ethernet connection is not possible, you can instead connect the IP camera to a hub on your subnet (isolated from your production environment), and use a subnet to connect the AWS Panorama Appliance Developer Kit to the AWS Cloud.

Network configuration for testing with production data (less secure)

If you need to test production data with the AWS Panorama Appliance Developer Kit, you can use a separate Virtual Local Area Network (VLAN) and restrict access to the VLAN only to the camera and the AWS Panorama Appliance Developer Kit. Use a firewall to limit incoming and outgoing traffic only to the AWS Cloud endpoints necessary for operation. This ensures there is a one-way communication only from the camera to the AWS Panorama Appliance Developer Kit, and isolates your test environment from any other VMS and any production environment.

Releases

The following table shows when features and software updates were released for the AWS Panorama service, software, and documentation. To ensure that you have access to all features, update your AWS Panorama Appliance (p. 27) to the latest software version. For more information on a release, see the linked topic.

update-history-change	update-history-description	update-history-date
Preview (p. 66)	AWS Panorama is now available by invitation in the US East (N. Virginia) and US West (Oregon) Regions. To apply to purchase an AWS Panorama Appliance Developer Kit, visit AWS Panorama.	December 1, 2020