AWS Ground Station

User Guide

AWS Ground Station: User Guide

The AWS Documentation website is getting a new look!

Try it now and let us know what you think. Switch to the new look >>

You can return to the original look by selecting English in the language selector above.

Table of Contents

What Is AWS Ground Station?

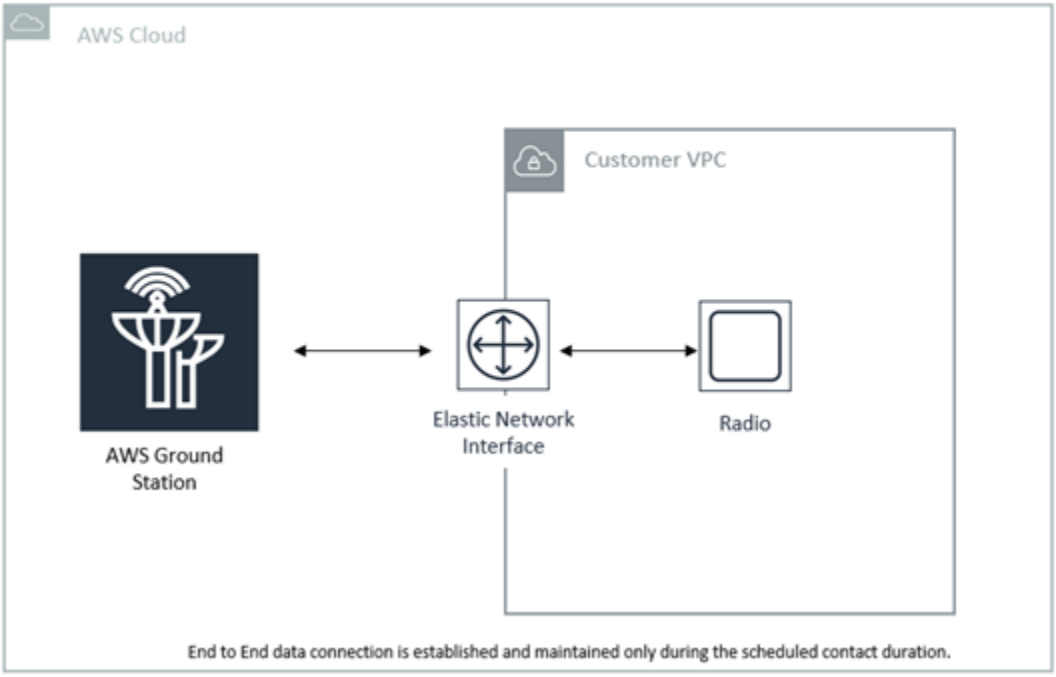
AWS Ground Station is a fully managed service that enables you to control satellite communications, process satellite data, and scale your satellite operations. This means that you no longer have to build or manage your own ground station infrastructure.

AWS Ground Station enables you to focus on innovating and rapidly experimenting with new applications that ingest satellite data and dynamically scale your server and storage use, rather than spend resources on operating and maintaining your own ground stations.

How AWS Ground Station Works

A satellite reservation is also known as a *contact*. Your satellite communicates with an AWS Ground Station antenna during contacts. You can reserve contacts through an API or through the AWS console by specifying location, time, and mission information. Your contact data can be streamed to and from an Amazon Elastic Compute Cloud (Amazon EC2) instance. From there, you can process the data or pass it into an Amazon S3 bucket, where you can use other AWS services like Amazon Machine Learning (Amazon ML) or Amazon Rekognition for post-processing. This ensures control over your data.

You can create extensible and reusable configuration resources so that you have control over how AWS Ground Station antennas are configured during your contacts. Using *mission profiles*, you can specify where data is coming from, what its format should be, and where to send it.



With AWS Ground Station you can access more than 125 services via satellite communications. Note the following:

- You can receive radio frequency (RF) data in X-band (8000 to 8500 MHz) or S-band (2200 to 2300 MHz) at bandwidths up to 54 MHz, called *narrowband*.
  - The X-Band data is down-converted to intermediate frequency (IF), digitized, and provided as a digital stream using RF-over-IP VITA 49 format.
  - The S-Band data is digitized and provided as a digital stream using RF-over-IP VITA 49 format.
- You can receive RF data in X-band (8000 to 8500 MHz) at bandwidths up to 500 MHz, called *wideband*.

  - The X-Band data is down-converted to IF, digitized, demodulated, decoded, and provided as a digital stream using Baseband VITA 49 format.
- You can transmit data in S-Band (2025 to 2120 MHz) at bandwidths up to 54 MHz.
  - The data is provided to the antenna as a digital stream using RF-over-IP VITA 49 format.
- You must run AWS Ground Station from the US East (Ohio) or US West (Oregon) Regions.
- Your EC2 instance must exist in the same AWS Region as the antenna.

Service Terms

You may only use the Services to store, retrieve, query, serve, and execute Your Content that is owned, licensed or lawfully obtained by you. As used in these Service Terms, (a) "Your Content" includes any "Company Content" and any "Customer Content" and (b) "AWS Content" includes "Amazon Properties." As part of the Services, you may be allowed to use certain software (including related documentation) provided by us or third-party licensors.

Important
This software is neither sold nor distributed to you and you may use it solely as part of the Services. You may not transfer it outside the Services without specific authorization to do so.

Getting Started with AWS Ground Station

AWS Ground Station enables you to command, control, and downlink data from your satellites.

With AWS Ground Station, you can schedule access to ground station antennas on a per-minute basis and pay only for the antenna time used. Incoming data is streamed to an AWS backend infrastructure where other AWS services can store or process it.

Topics

Setting Up AWS Ground Station

Before you start using AWS Ground Station, you need to know what AWS Identity and Access Management (IAM) permissions you need, and what space vehicle credentials to provide.

Step 1: Sign Up for AWS

To use AWS Ground Station, you need an AWS account. If you already have an AWS account, skip to the section called "Step 2: Create an IAM User in Your AWS Account" (p. 4).

To sign up for an AWS account

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

   Note If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then, choose **Create a new AWS account**.
2. Follow the instructions. Part of the sign-up procedure involves receiving a phone call and entering a verification code using the phone keypad.

Step 2: Create an IAM User in Your AWS Account

To use AWS Ground Station, you need to create an IAM user in your AWS account with specific permissions.

To create an IAM user in your AWS account

1. In Creating an IAM User in Your AWS Account, follow the instructions under **Creating IAM Users (Console)**.
2. In the AWS Management Console, choose **Users** and in the **User name** list, choose the new user that you just created.
3. On the **Permissions** tab, choose **Add permissions**.

4. On the next page, choose **Attach existing policies directly**.
5. Find the **Groundstation_general_access** policy.
6. In the edit pane of the **Permissions** tab, edit the JSON with the following values:

   - For Console General Access, set **Action** to **groundstation:*** as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "groundstation:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

   - For Read-only, set **Action** to **groundstation:Get***, **groundstation:List***, and **groundstation:Describe*** as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "groundstation:Get*",
        "groundstation:List*",
        "groundstation:Describe*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

   - For additional security through multifactor authentication, set **Action** to **groundstation:***, and **Condition/Bool** to **aws:MultiFactorAuthPresent:true** as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "groundstation:*",
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "aws:MultiFactorAuthPresent": true
                }
            }
        }
    ]
}
```

For more information about creating IAM users and attaching policies, see the *IAM User Guide*.

Step 3: Customer Onboarding

To complete registration for your AWS Ground Station account, see the AWS Ground Station webpage to provide onboarding details. The AWS Ground Station team will work with you to onboard your satellites to the service.

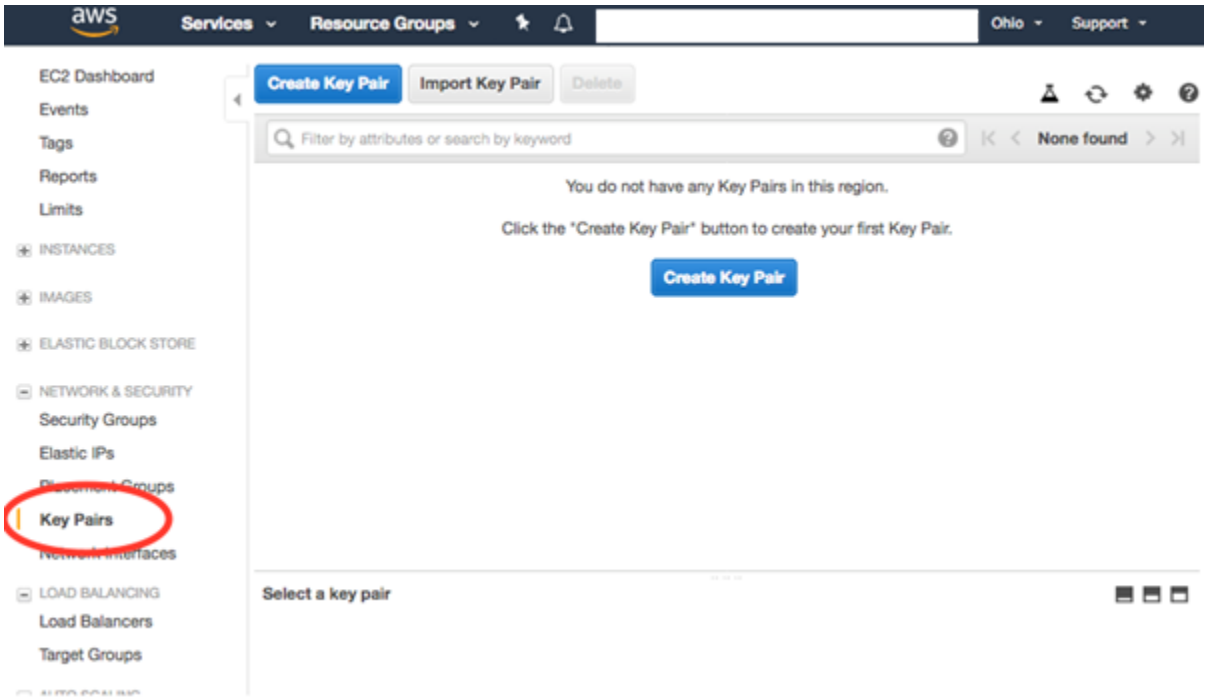
Step 1: Set Up Your VPC

The full setup of a VPC is beyond the scope of this guide. If you don't have an existing VPC that is already customized, you can use the default VPC that is created in your AWS account. We recommend adding a Linux bastion to your VPC so that you can SSH into your Amazon EC2 instances without attaching a public IP address. For more information about configuring a Linux bastion in your VPC, see http://aws.amazon.com/quickstart/architecture/linux-bastion/.

Step 2: Create EC2 SSH Key Pair

If you do not already have one, create a new key pair in the Amazon EC2 console for each AWS Region where you plan to receive data:

1. Choose one of the following AWS Regions: US East (Ohio) or US West (Oregon). You need to create a key pair for every AWS Region in which you plan to reserve contacts.
2. Choose **Services** > **EC2** > **Network & Security** > **Key Pairs**, and then choose **Create Key Pair**.
3. Enter a friendly name like **bastion-key-*<region>*** (for example, bastion-key-useast2).
4. Save the private key, make it accessible to your ssh utility of choice, and set the ownership/permissions as needed (for example, chmod 600 *<key.pem>*).
5. Repeat for the other AWS Region if needed.



Step 3: Customize and Run the AWS CloudFormation Template

As you read the description in this topic, know that we do not expect you to figure this out on your own. For support, see the AWS Ground Station website.

After you onboard (p. 6) your satellite, you need to define mission profiles and create instances to process or push data streams from or to your satellite. To assist you with this process, we provide the following example AWS CloudFormation templates as a base to create the resources for your use case. The following example templates build on one another so you can see the additions and updates required to add new resources to each:

- the section called "Downlink Only Template" (p. 7)
- the section called "Downlink Demod/Decode Template" (p. 10)
- the section called "Downlink Demod/Decode and Uplink Template" (p. 10)
- the section called "Downlink Demod/Decode and Uplink Echo Template" (p. 11)

Today, you can configure multiple streams of data per contact to flow into your VPC. These data streams are available in two different formats, depending on the frequency and bandwidth of the RF signal you are receiving during your contact. VITA-49 Signal/IP data streams can be used for S-Band and X-Band signals up to 54 MHz in bandwidth. Alternatively, VITA-49 Uncoded Frames/IP can be used for only X-Band signals up to 600 MHz in bandwidth.

After you have been onboarded (p. 6), you can access the following Amazon S3 bucket to download the example templates in s3://groundstation-customer-cf/ where you can find the templates mentioned in the following sections.

Downlink Only Template

The AWS CloudFormation template named `DownlinkOnlyCustomerTemplate.yml` defines an AWS CloudFormation stack that provides all the resources necessary for a single downlink stream. This is a good starting point for building out more complex use cases as well. You can download the template using the following AWS CLI command:

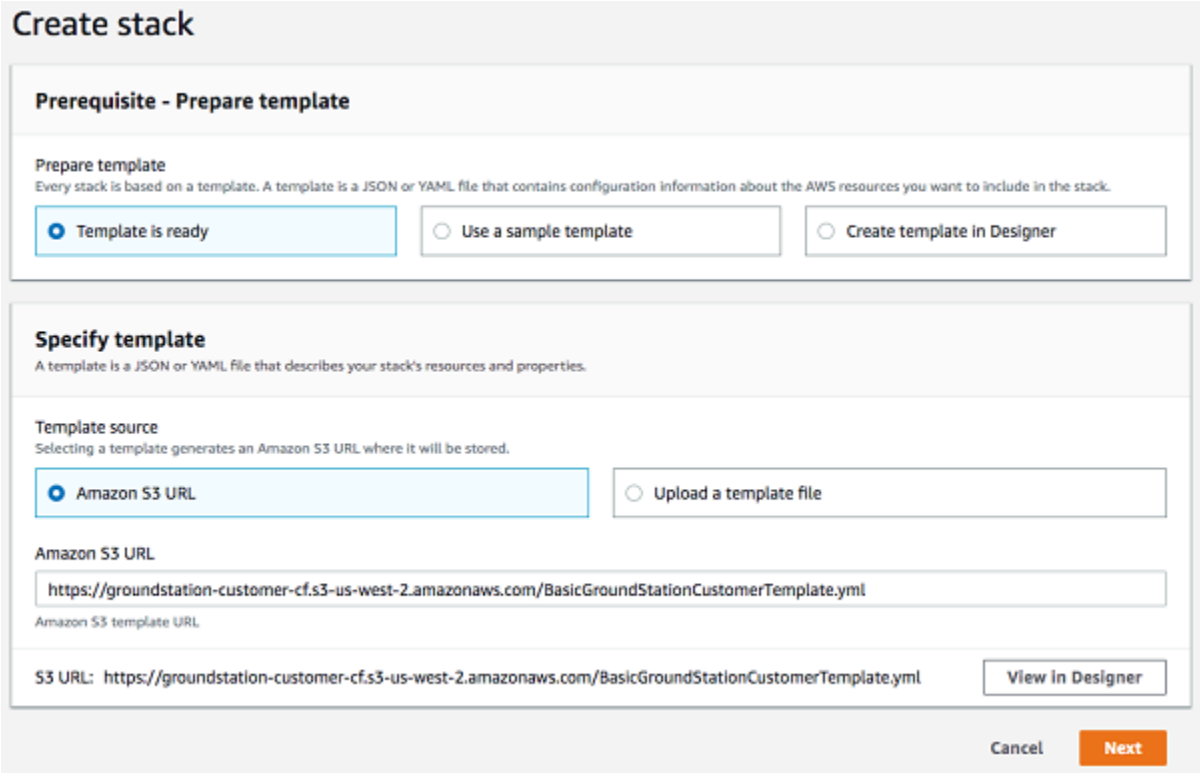
```
aws s3 cp s3://groundstation-customer-cf/DownlinkOnlyCustomerTemplate.yml
```

The template can be viewed and downloaded in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-customer-cf/
DownlinkOnlyCustomerTemplate.yml
```

The template can be specified directly in AWS CloudFormation by using the following link:

```
https://groundstation-customer-cf.s3-us-west-2.amazonaws.com/
DownlinkOnlyCustomerTemplate.yml
```



What resources does the template define?

The template defines the following resources:

- **Dataflow Endpoint Group -** The AWS Ground Station dataflow endpoint group (p. 13) that defines the endpoints used to send/receive data to/from your satellite. As part of the Dataflow Endpoint Group creation AWS Ground Station creates an elastic network interface in your account to stream data.
- **Dataflow Endpoint Security Group -** The security group that the elastic network interface created by AWS Ground Station belongs to. By default, this security group allows AWS Ground Station to stream traffic to any IP address in your VPC. This can be modified such that traffic is limited to a specific set of IP addresses.
- **Data Delivery Service Role -** AWS Ground Station assumes this role to create/delete ENIs in your account in order to stream data.
- **Tracking Config -** The AWS Ground Station tracking config (p. 15) that defines how the antenna system tracks your satellite as it moves through the sky.
- **Antenna Downlink Config -** The AWS Ground Station antenna downlink config (p. 16) that defines the frequency spectrum used to downlink data from your satellite.
- **Downlink Endpoint Config -** The AWS Ground Station dataflow endpoint config (p. 15) that defines the endpoint used to downlink data from your satellite.
- **Mission Profile -** The AWS Ground Station mission profile (p. 18) that groups the Tracking, Antenna Downlink, and Downlink Endpoint configurations to define how to uplink to and downlink data from your satellite.
- **Receiver Instance -** The Amazon EC2 instance that will send/receive data to/from your satellite using AWS Ground Station.
  - **Instance Security Group -** The security group for your Amazon EC2 instance.
  - **Instance Role -** The role for your Amazon EC2 instance.

- **Instance Profile -** The instance profile for your Amazon EC2 instance.
- **Cluster Placement Group -** The placement group in which your Amazon EC2 instance is placed.

- **Receiver Instance Network Interface -** An elastic network interface that provides a fixed IP address for AWS Ground Station to connect to. This is attached to the receiver instance on `eth1`.

What changes should I make to the AWS CloudFormation template?

You need to change the AWS CloudFormation template to include correct frequencies for your satellites. This is defined in the `spectrumConfig` section of the antenna downlink config.

The antenna downlink config (shown following) has a lower bound on carrier bandwidth and adjusts this automatically if it's lower than 1 MHz. If it's adjusted automatically, a 10MHz offset is added/subtracted to/from the center frequency.

```
spectrumConfig:

    bandwidth:
     units: "MHz"
     value: 10.0
    centerFrequency:
     units: "MHz"
     value: 2250.0
    polarization: "RIGHT_HAND"
```

What is on the Amazon EC2 instance that the AWS CloudFormation template creates?

The AWS CloudFormation template defines a startup script (`UserData`) that installs RT Logic Data Defender software and a Python script to automatically configure it. Data Defender ensures bit-for-bit accuracy of the data streams between your Amazon EC2 instance and the antenna system using DTLS.

The Python configuration script provides a method of configuring the Data Defender streams using AWS CloudFormation. It configures the streams based on a JSON file created by the startup script of the Amazon EC2 instance. The following JSON file shows the single downlink stream defined by the AWS CloudFormation template's parameters:

```
{
  "ddx_streams": [
    {
      "streamName": "${DownlinkStreamName}",
      "maximumWanRate": 4000000000,
      "lanConfigDevice": "lo",
      "lanConfigPort": 50000,
      "wanConfigDevice": "eth1",
      "wanConfigPort": ${DownlinkPort},
      "isUplink": false
    }
  ]
}
```

Important
The stream names and ports use AWS CloudFormation's `Fn::Sub` function to populate the names of the streams and port numbers based on the values provided in the template's parameters.

Downlink Demod/Decode Template

The AWS CloudFormation template named `DownlinkDemodDecodeCustomerTemplate.yml` is based on the the section called "Downlink Only Template" (p. 7) from the previous section. It provides all the resources necessary for a single downlink stream that is demodulated and decoded by the AWS Ground Station service based on the settings you provide. You can download the template using the following AWS CLI command:

```
aws s3 cp s3://groundstation-customer-cf/DownlinkDemodDecodeCustomerTemplate.yml
```

The template can be viewed and downloaded in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-customer-cf/
DownlinkDemodDecodeCustomerTemplate.yml
```

The template can be specified directly in AWS CloudFormation by using the following link:

```
https://groundstation-customer-cf.s3-us-west-2.amazonaws.com/
DownlinkDemodDecodeCustomerTemplate.yml
```

What resources does the template define?

The downlink demod/decode template does not define any new resources on top of those defined in the downlink only template. However, it does add a new property called `antennaDownlinkDemodDecodeConfig` in the antenna downlink config (p. 16) properties.

What changes should I make to the AWS CloudFormation template?

In addition to the changes mentioned in the downlink only template (p. 7) section, the demodulation and decoding settings in the template need to be modified to work with the modulation and encoding scheme of the satellite you want to downlink from. These settings are defined in the `antennaDownlinkDemodDecodeConfig` section of the antenna downlink config. For more information, see the section called "Antenna Downlink Demod Decode Config" (p. 17).

Downlink Demod/Decode and Uplink Template

The AWS CloudFormation template named `DownlinkDemodDecodeAndUplinkCustomerTemplate.yml` is based on the the section called "Downlink Demod/Decode Template" (p. 10) from the previous section. It provides all the resources necessary for a single downlink stream and a single uplink stream. You can download the template using the following AWS CLI command:

```
aws s3 cp s3://groundstation-customer-cf/DownlinkDemodDecodeAndUplinkCustomerTemplate.yml
```

The template can be viewed and downloaded in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-customer-cf/
DownlinkDemodDecodeAndUplinkCustomerTemplate.yml
```

The template can be specified directly in AWS CloudFormation by using the following link:

```
https://groundstation-customer-cf.s3-us-west-2.amazonaws.com/
DownlinkDemodDecodeAndUplinkCustomerTemplate.yml
```

What resources does the template define?

The downlink demod/decode and uplink template defines the following resources in addition to the resources defined in the downlink demod/decode template from the previous section:

- **Downlink Demod/Decode –** This template includes all of the resources defined in the downlink demod/decode template.
- **Security Groups –** This template updates the properties of the security groups to include ingress and egress for the uplink stream's port.
- **Data Defender Streams –** This template adds a new entry for the uplink stream in the stream definition JSON blob contained in the Amazon EC2 instance's `UserData` property. Note that the `isUplink` property in the uplink stream's definition is set to `true`, whereas the downlink stream is set to `false`.
- **Dataflow Endpoint Group –** This template updates the dataflow endpoint group (p. 13) so that it contains a new endpoint for the uplink stream.
- **Antenna Uplink Config –** The AWS Ground Station antenna uplink config (p. 16) defines the frequency spectrum used to uplink data to your satellite.
- **Uplink Endpoint Config –** The AWS Ground Station dataflow endpoint config (p. 15) defines the endpoint used to uplink data to your satellite.

What changes should I make to the AWS CloudFormation template?

In addition to the changes mentioned in the the section called "Downlink Demod/Decode Template" (p. 10) section, the antenna uplink config needs to be updated so that the `spectrumConfig` property has the correct `centerFrequency` and `polarization` for your satellite.

The antenna uplink config also includes the target effective isotropic radiated power (EIRP) for your uplinks. This value should be set to an appropriate level such that the signal is strong enough for uplinks to your satellites.

```
spectrumConfig:

        centerFrequency:
          units: "MHz"
          value: 2072.5
        polarization: "RIGHT_HAND"
          targetEirp:
          units: "dBW"
          value: 20.0
```

Downlink Demod/Decode and Uplink Echo Template

The AWS CloudFormation template named `DownlinkDemodDecodeAndUplinkEchoCustomerTemplate.yml` is based on the the section called "Downlink Demod/Decode and Uplink Template" (p. 10) from the previous section. It provides all the resources necessary for a single downlink stream, a single uplink stream, and an uplink echo stream. You can download the template using the following AWS CLI command:

```
aws s3 cp s3://groundstation-customer-cf/
DownlinkDemodDecodeAndUplinkEchoCustomerTemplate.yml
```

The template can be viewed and downloaded in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-customer-cf/
DownlinkDemodDecodeAndUplinkEchoCustomerTemplate.yml
```

You can specify the template directly in AWS CloudFormation using the following link:

```
https://groundstation-customer-cf.s3-us-west-2.amazonaws.com/
DownlinkDemodDecodeAndUplinkEchoCustomerTemplate.yml
```

What resources does the template define?

The downlink demod/decode and uplink echo template defines the following resources in addition to the resources defined in the the section called “Downlink Demod/Decode and Uplink Template” (p. 10) from the previous section:

- **Downlink Demod/Decode and Uplink –** This template includes all of the resources defined in the downlink demod/decode and uplink template.
- **Security Groups –** This template updates the properties of the security groups to include ingress and egress for the uplink echo stream's port.
- **Data Defender Streams –** This template adds a new entry for the uplink echo stream in the stream definition JSON blob contained in the Amazon EC2 instance's `UserData` property.
- **Dataflow Endpoint Group –** This template updates the dataflow endpoint group (p. 13) so that it contains a new endpoint for the uplink echo stream.
- **Uplink Echo Config –** The AWS Ground Station antenna uplink echo config (p. 17) specifies the uplink stream from which to echo back to your Amazon EC2 instance.
- **Uplink Echo Endpoint Config –** The AWS Ground Station dataflow endpoint config (p. 15) defines the endpoint used to stream data from the uplink echo back to your Amazon EC2 instance.

What changes should I make to the AWS CloudFormation template?

The downlink demod/decode and uplink echo template doesn't require any changes in addition to the changes mentioned in the the section called “Downlink Demod/Decode and Uplink Template” (p. 10) section.

Step 4: Install and Configure FE Processor/Radio

The Amazon EC2 instance defined in the AWS CloudFormation template does not have a Front End (FE) processor or software defined modem (SDM) installed by default. You need to install an FE processor or SDM in order to process the VITA-49 packets streamed to/from the AWS Ground Station antenna system.

How you install and configure your FE processor or SDM depends on which FE processor or SDM you are using. Installation of an FE processor or SDM is beyond the scope of this user guide.

> **Important**
> It's a best practice to run your FE processor or SDM on the instances created by the AWS CloudFormation template to ensure the benefits of the DTLS data streams to/from Data Defender.

Dataflow Endpoint Groups

Dataflow endpoints define the location where you want the data to be streamed to or from during contacts. Endpoints contain a name that you choose to identify this type of endpoint. Dataflow endpoint names do not need to be unique, but two configs with the same name are interchangeable. Dataflow endpoint configs use the name to choose which dataflow endpoint to use during contacts.

The endpoint list address consists of the following:

- `name` - IP address of this dataflow endpoint.
- `port` - The port to connect to.

The security details of an endpoint consist of the following:

- `roleArn` - The Amazon Resource Name (ARN) to a role that AWS Ground Station needs in order to connect streams to your instances.
- `securityGroupIds` - The security groups to attach to the elastic network interfaces.
- `subnetIds` - A list of subnets where AWS Ground Station places elastic network interfaces to send streams to your instances

Dataflow endpoints are always created as part of a *dataflow endpoint group*. By including multiple dataflow endpoints in a group, you are asserting that the specified endpoints can all be used together during a single contact. For example, if a contact needs to send data to three separate dataflow endpoints, you must have three endpoints in a single dataflow endpoint group that match the dataflow endpoint configs in your mission profile.

When one or more resources in a dataflow endpoint group is in use for a contact, the entire group is reserved for the duration of that contact.

You may execute multiple contacts at a time, but those contacts must be executed on different dataflow endpoint groups:

```
{
    "endpointDetails": [
        {
            "endpoint": {
                "address": {
                    "name": "192.168.1.1",
                    "port": 55888
                    },
                "name": "DataflowEndpoint1",
            },
            "securityDetails": {
                "roleArn": "string",
                "securityGroupIds": [ "string" ],
                "subnetIds": [ "string" ]
            }
        },
        {
            "endpoint": {
                "address": {
                    "name": "192.168.1.1",
                    "port": 55889
                    },
```

```
                "name": "DataflowEndpoint2",
            },
            "securityDetails": {
                "roleArn": "string",
                "securityGroupIds": [ "string" ],
                "subnetIds": [ "string" ]
            }
        }
    ]
}
```

Configs

Configs are resources that AWS Ground Station uses to define the parameters for each aspect of your contact. Add the configs you want to a mission profile, and then that mission profile will be used when executing the contact. You can define several different types of configs.

Dataflow Endpoint Config

You can use dataflow endpoint configs to specify which dataflow endpoints you want to use during contacts. The only parameter in the config itself is the dataflow endpoint name. The following example tells the contact to use a dataflow endpoint with the name `DataflowEndpoint1` from any dataflow endpoint group.

You can use the following example code to create multiple dataflow endpoint groups across AWS Regions and reuse the same `Config` to execute contacts.

```
{
    "configData": {
        "dataflowEndpointConfig": {
            "dataflowEndpointName": "DataflowEndpoint1"
        }
    },
    "name": "MyDataflowEndpointConfig"
}
```

Tracking Config

You can use tracking configs in the mission profile to determine whether autotrack should be enabled during your contacts. This config has a single parameter: `autotrack`. The `autotrack` parameter can have the following values:

- `REQUIRED` - Autotrack is required for your contacts.
- `PREFERRED` - Autotrack is preferred for contacts, but contacts can still be executed without autotrack.
- `REMOVED` - No autotrack should be used for your contacts.

The following is an example tracking config:

```
{
    "configData": {
        "trackingConfig": {
            "autotrack": "string"
        },
    },
    "name": "MyTrackingConfig"
}
```

Antenna Downlink Config

You can use antenna downlink configs to configure the antenna during your contact. They consist of a spectral config that specifies the `bandwidth`, `frequency`, and `polarization` that should be used during your antenna downlink. If your downlink use case requires demodulation or decode, see the the section called “Antenna Downlink Demod Decode Config” (p. 17).

The following is an example antenna downlink config.

```
{
    "antennaDownlinkConfig": {
        "spectrumConfig": {
            "bandwidth": {
                "units": "MHz",
                "value": 25
            },
            "centerFrequency": {
                "units": "MHz",
                "value": 8212.5
            },
            "polarization": "RIGHT_HAND"
        }
    },
    "name": "MyAntennaDownlinkConfig"
}
```

Antenna Uplink Config

You can use antenna uplink configs to configure the antenna during your uplink contact. They consist of a spectral config with `frequency`, `polarization`, and `targetEirp`. For information about how to configure uplink echo, see the section called “Antenna Uplink Echo Config” (p. 17).

The following is an example antenna uplink config.

```
{
    "configData": {
        "antennaUplinkConfig": {
            "spectrumConfig": {
                "centerFrequency": {
                    "units": "MHz",
                    "value": 2091
                },
                "polarization": "RIGHT_HAND"
            },
            "targetEirp": {
                "units": "dBW",
                "value": 0
            }
        }
    },
    "name": "MyAntennaDownlinkConfig"
}
```

Antenna Uplink Echo Config

Uplink echo configs tell the antenna how to execute uplink echo. This echoes commands sent by the antenna back to your dataflow endpoint. Uplink echo config contains the ARN of an uplink config. The antenna uses the parameters from the uplink config pointed to by the ARN when executing uplink echo.

The following is an example antenna uplink echo config.

```
{
    "configData": {
        "uplinkEchoConfig": {
            "antennaUplinkConfigArn": "arn:aws:groundstation:us-east-2:123456789012:config/
antenna-uplink/11111111-2222-3333-4444-555555555555",
            "enabled": true
        }
    },
    "name": "MyAntennaUplinkEchoConfig"
}
```

Antenna Downlink Demod Decode Config

Antenna downlink demod decode configs are a more complex and customizable config type that you can use to execute downlink contacts with demod or decode. If you're interested in executing these types of contacts, contact the AWS Ground Station team. We'll help you define the right config and mission profile for your use case.

Mission Profiles

Mission profiles contain configs and parameters for how contacts are executed. When you reserve a contact or search for available contacts, you supply the mission profile that you intend to use. Mission profiles bring all of your configs together and define where data will go during your contact.

Aside from tracking configs (p. 15), all configs are contained in the dataflow edges field of the mission profile. A single data flow edge is a list of two ARNs—the first is the *from* config and the second is the *to* config. By specifying a dataflow edge between two configs, you are telling the ground station system how your data should flow during a contact. Tracking configs are not used as part of a dataflow edge, but are specified as a separate field.

The name field of the mission profile helps distinguish between the mission profiles that you create.

The following is an example mission profile for the simplest downlink use case.

```
 {
    "contactPostPassDurationSecond": 60,
    "contactPrePassDurationSeconds": 60,
    "dataflowEdges": [
        [
            "arn:aws:groundstation:us-east-2:123456789012:config/antenna-
downlink/11111111-2222-3333-4444-555555555555",
            "arn:aws:groundstation:us-east-2:123456789012:config/dataflow-
endpoint/11111111-2222-3333-4444-555555555555"
        ]
    ],
    "minimumViableContactDurationSeconds": 120,
    "name": "MySimpleAntennaDownlinkMissionProfile",
    "trackingConfigArn": "arn:aws:groundstation:us-west-2:123456789012:config/
tracking/11111111-2222-3333-4444-555555555555"
}
```

The following is an example mission profile for a contact containing downlink, uplink, and uplink echo.

```
{
    "contactPostPassDurationSecond": 60,
    "contactPrePassDurationSeconds": 60,
    "dataflowEdges": [
        [
            "arn:aws:groundstation:us-east-2:123456789012:config/antenna-
downlink/11111111-2222-3333-4444-555555555555",
            "arn:aws:groundstation:us-east-2:123456789012:config/dataflow-
endpoint/11111111-2222-3333-4444-555555555555"
        ],
        [
            "arn:aws:groundstation:us-east-2:123456789012:config/antenana-
uplink/11111111-2222-3333-4444-555555555555",
            "arn:aws:groundstation:us-east-2:123456789012:config/dataflow-
endpoint/31111111-2222-3333-4444-555555555555"
        ],
        [
            "arn:aws:groundstation:us-east-2:123456789012:config/uplink-
echo/11111111-2222-3333-4444-555555555555",
            "arn:aws:groundstation:us-east-2:123456789012:config/dataflow-
endpoint/11111111-2222-3333-4444-555555555555"
```

```
        ]
    ],
    "minimumViableContactDurationSeconds": 120,
    "name": "MyDownlinkUplinkEchoMissionProfile",
    "trackingConfigArn": "arn:aws:groundstation:us-west-2:123456789012:config/
tracking/11111111-2222-3333-4444-555555555555"
}
```

Using the AWS Ground Station Console

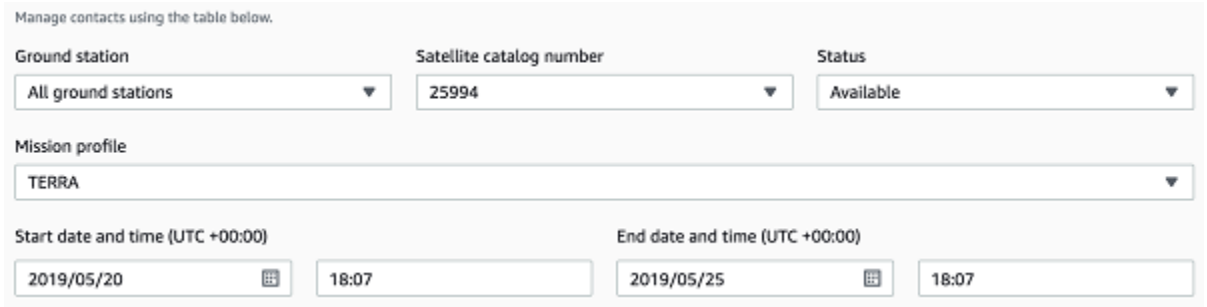
In the AWS Ground Station console, you can enter satellite data, identify antenna locations, communicate, and schedule antenna time for selected satellites. You can review, cancel, and reschedule contact reservations up to eight days prior to scheduled time.

Listing and Reserving Contacts

In the AWS Ground Station console, you can schedule, review, and cancel contacts with your satellites. You can also view the details of your reserved minutes pricing plan if you use the AWS Ground Station reserved minutes pricing model.

1. Open the AWS Ground Station console and choose **Reserve contacts now**.

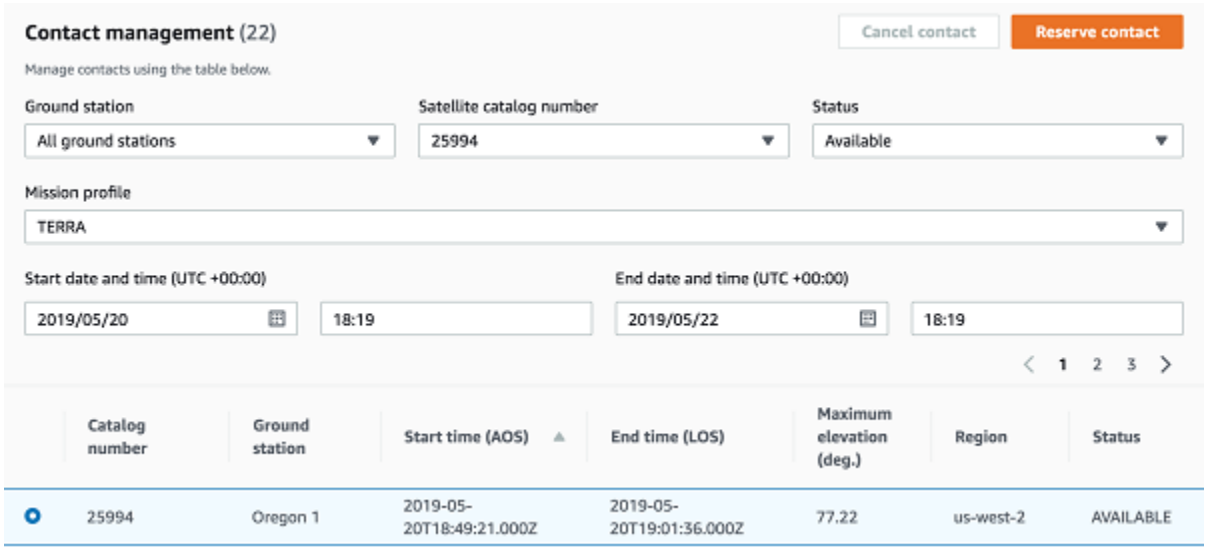
2. Use the table to select the parameters you would like to search with.



   You can filter contacts based on Status (**Available**, **Reserved**, or **Completed**) and time range.

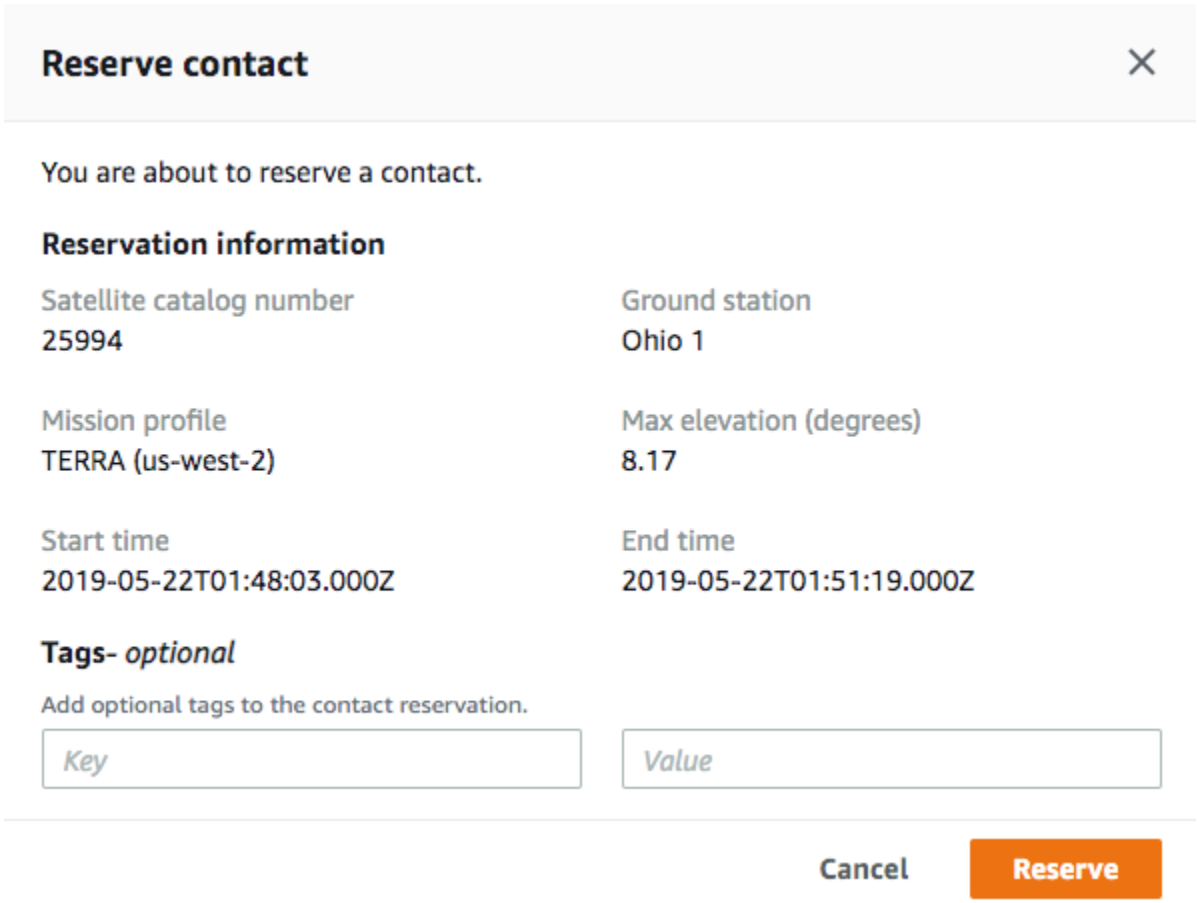
   Note
   If you are searching for **Available** contacts, you must select a mission profile. When a mission profile is selected for the **Scheduled** or **Completed** statuses, the returned contacts *won't* be filtered by mission profile.

3. Choose a contact that meets your requirements and then choose **Reserve contact**.



4. To confirm your contact, on the next page, choose **Reserve**.

   - To add tags to your contact, use the **Tags** fields.



Result: AWS Ground Station uses the configuration data from your mission profile to execute a contact at the specified ground station.

AWS Ground Station Security

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

Use the following topics to learn how to secure your AWS Ground Station resources.

Topics

Authentication and Access Control for AWS Ground Station

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 Ground Station resources. IAM is a feature of your AWS account offered at no additional charge.

Topics

Audience

Authentication and access control matters to you in different ways, depending on the work you do in AWS Ground Station.

Service user – If you use the AWS Ground Station service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Ground Station features to do your work, you might need additional permissions. Understanding access control can help you request the right permissions from your administrator. If you cannot access a feature in AWS Ground Station, see Troubleshooting AWS Ground Station Authentication and Access Control (p. 33).

Service administrator – If you're in charge of AWS Ground Station resources at your company, you probably have full access to AWS Ground Station. It's your job to determine which AWS Ground Station 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 authentication and controlling access. To learn more about how your company can use IAM with AWS Ground Station, see How AWS Ground Station Works with IAM (p. 27).

IAM administrator – If you're an IAM administrator, you already understand the concepts of authentication and access control. But you want to learn details about how you can write policies to control access to AWS Ground Station. To view example AWS Ground Station identity-based policies that you can use in IAM, see AWS Ground Station Identity-Based Policy Examples (p. 30).

Authentication

Authentication is how you sign in to AWS using your credentials.

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 sign in to the AWS Management Console or access AWS programmatically. 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*. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

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 entity 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 authenticate from the AWS Management Console as an IAM user, you must sign in with your user name and password. To learn more about signing in to the console, see How IAM Users Sign In to AWS in the *IAM User Guide*. To be authenticated from the AWS CLI or AWS API, you must provide your IAM user access key ID and secret key. 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 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 a collection of IAM users. 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 *AdminIAM* 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 entity 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 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*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS 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, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

Controlling 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 entity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. 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 GetUser action. A user with that policy can get user 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, role, or group. These policies control what actions that identity 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*.

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. You can set a permissions boundary for an entity. That entity can then perform only the actions that are allowed by both its identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role as the principal 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 the 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 About Service Control Policies 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 permissions for a session come from identity-based policies for the IAM entity (user or role) used to create the session and the session policy. Permissions can also come from a resource-based policy. 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*.

Learn More

For more information about authentication and access control for AWS Ground Station, continue to the following pages:

- How AWS Ground Station Works with IAM (p. 27)
- Troubleshooting AWS Ground Station Authentication and Access Control (p. 33)

How AWS Ground Station Works with IAM

Before you use IAM to control access to AWS Ground Station, you should understand what IAM features are available to use with AWS Ground Station. To get a high-level view of how AWS Ground Station and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

AWS Ground Station Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. AWS Ground Station supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

AWS Ground Station Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions in AWS Ground Station use the following prefix before the action: `groundstation`. For example: `groundstation:Get*`, `groundstation:List*`, `groundstation:Describe*` (for all AWS Ground Station actions). For a list of the actions, see the Actions Defined by AWS Ground Station.

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

```
"Action": "groundstation:Describe*"
```

The following table describes actions which are common for console access:

Action	Description
`CancelContact`	Grants permission to cancel a contact
`DescribeContact`	Grants permission to describe a contact
`ListContacts`	Grants permission to return a list of contacts
`ReserveContact`	Grants permission to reserve a contact

To see a list of AWS Ground Station actions, see the AWS Ground Station API Reference.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

The AWS Ground Station `Config` resource has the following ARN:

```
arn:${Partition}:groundstation:${Region}:${AccountID}:config/${configType}/${configId}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the `11111111-2222-3333-4444-555555555555` `Config` in your statement, use the following ARN:

```
"Resource": "arn:aws:groundstation:us-east-2:123456789012:config/
DecodeConfig/11111111-2222-3333-4444-555555555555"
```

To specify all `Config` objects that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:groundstation:us-east-2:123456789012:config/*"
```

Some AWS Ground Station actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

Many AWS Ground Station API actions involve multiple resources. For example, `CreateConfig` can create an AWS Ground Station `Config` across multiple satellites, so an IAM user must have permissions to use the `Config` and the contact. To specify multiple resources in a single statement, separate their ARNs with commas.

```
"Resource": [
               "arn:aws:groundstation:us-west-2:123456789012:config/
satellite/11111111-2222-3333-4444-555555555555",
               "arn:aws:groundstation:us-west-2:123456789012:config/
satellite/21111111-2222-3333-4444-555555555555"
```

The following table summarizes how to create resources with AWS Ground Station:

Action	Description
`CreateConfig`	Grants permission to create a Config
`CreateDataflowEndpointGroup`	Grants permission to create a dataflow endpoint group
`CreateMissionProfile`	Grants permission to create a mission profile

A `Contact` is another common resource in AWS Ground Station, which has a resource ARN such as,

```
"arn:aws:groundstation:us-west-2:123456789012:contact/11111111-2222-3333-4444-555555555555"
```

The following table summarizes how to update other resources:

Action	Description
`UpdateConfig`	Grants permission to update a Config

Action	Description
`UpdateMissionProfile`	Grants permission to update a mission profile

To see a list of AWS Ground Station resource types and their ARNs, see Resources Defined by AWS Ground Station in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS Ground Station.

Condition Keys

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. AWS Ground Station defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

> **Note**
> Do not use the `aws:SourceIp` AWS global condition key with AWS CloudFormation. AWS CloudFormation provisions resources by using its own IP address, not the IP address of the originating request. For example, AWS CloudFormation makes requests from its IP address to launch an Amazon EC2 instance or to create an Amazon S3 bucket. It does not use the IP address from the `CreateStack` operation or the `aws cloudformation create-stack` command. Nor does it use the IP address of the person who makes the call.

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

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

All AWS Ground Station actions support the `aws:RequestedRegion` and `groundstation:Region` condition keys. For more information, see Example: Restricting Access to a Specific Region.

To see a list of AWS Ground Station condition keys, see Condition Keys for AWS Ground Station in the *IAM User Guide*. To learn with which actions and resources you can use a condition key, see Actions Defined by AWS Ground Station.

Examples

To view examples of AWS Ground Station identity-based policies, see AWS Ground Station Identity-Based Policy Examples (p. 30).

AWS Ground Station Resource-Based Policies

AWS Ground Station does not support resource-based policies. To view an example of a detailed resource-based policy page, see https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Authorization Based on AWS Ground Station Tags

You can attach tags to AWS Ground Station resources or pass tags in a request to AWS Ground Station. To control access based on tags, you provide tag information in the condition element of a policy using

the `groundstation:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing AWS Ground Station ConfigIds Based on Tags (p. 32).

AWS Ground Station IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

AWS Ground Station currently does not support service-linked roles.

Using Temporary Credentials with AWS Ground Station

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

AWS Ground Station supports using temporary credentials.

AWS Ground Station Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify AWS Ground Station resources. They also can't perform tasks using the AWS Ground Station console, CLI, or 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*.

The following example policy grants Console General Access to AWS Ground Station.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "groundstation:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

The following example policy grants read-only access to AWS Ground Station.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "groundstation:Get*",
        "groundstation:List*",
        "groundstation:Describe*"
      ],
```

```
      "Resource": [
        "*"
      ]
    }
  ]
}
```

For more information about writing IAM policies, see *IAM Policies* 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 Ground Station resources in your account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using AWS Ground Station 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 Ground Station Console

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

To ensure that those entities can still use the AWS Ground Station console, also attach the following AWS managed policy to the user. For more information, see Adding Permissions to a User in the *IAM User Guide*:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "groundstation:Get*",
        "groundstation:List*",
        "groundstation:Describe*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, you need only the permissions that match the API operation that you're trying to perform.

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": "*"
           }
       ]
   }
```

Viewing AWS Ground Station ConfigIds Based on Tags

You can use conditions in your identity-based policy to control access to AWS Ground Station resources based on tags. This example shows how you might create a policy that allows viewing a `Config`.

However, permission is granted only if the `configId` tag `Owner` has the value of that user's user name. This policy also grants the permissions necessary to complete this action on the console.

```
{
    "Sid": "VisualEditor1",
    "Effect": "Allow",
    "Action": [
        "groundstation:GetConfig"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "groundstation:ResourceTag/Owner": "${aws:username}"
        }
    }
}
```

You can attach this policy to the IAM users in your account. If a user named `richard-roe` attempts to view an AWS Ground Station `configId`, the `configId` must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise he is denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case-sensitive. For more information, see IAM JSON Policy Elements: condition in the *IAM User Guide*.

Troubleshooting AWS Ground Station Authentication and Access Control

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

Topics

I am not authorized to perform an action in AWS Ground Station

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 `Config` but does not have `groundstation:Get*` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 groundstation:Get* on resource: my-example-config
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-example-config` resource using the `groundstation:Get*` 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 Ground Station.

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 Ground Station. 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, `AKIAIOSFODNN7EX`) and a secret access key (for example, `wJalrXUtnFEMI/K7M/bPxRfY`). 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 Ground Station

To allow others to access AWS Ground Station, 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 Ground Station.

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 Ground Station 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 Ground Station supports these features, see How AWS Ground Station Works with IAM (p. 27).
- To learn how to provide access to your resources across AWS accounts that you own, see Creating Your First IAM Delegated User and Group 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*.

Logging AWS Ground Station API Calls with AWS CloudTrail

AWS Ground Station is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Ground Station. CloudTrail captures all API calls for AWS Ground Station as events. The calls captured include calls from the AWS Ground Station console and code calls to the AWS Ground Station API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Ground Station. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Ground Station, the IP address from which the request was made, who made the request, when it was made, and additional details.

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

AWS Ground Station Information in CloudTrail

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

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

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS Ground Station actions are logged by CloudTrail and are documented in the AWS Ground Station API Reference. For example, calls to the `ReserveContact`, `CancelContact` and `ListConfigs` actions generate entries in the CloudTrail log files.

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

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding AWS Ground Station Log File Entries

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

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

Example: ReserveContact


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINICIPLE_ID",
        "arn": "arn:aws:sts::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-05-15T21:11:59Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINICIPLE_ID",
                "arn": "arn:aws:iam::123456789012:role/Alice",
                "accountId": "123456789012",
                "userName": "Alice"
            }
        }
    },
    "eventTime": "2019-05-15T21:14:37Z",
    "eventSource": "groundstation.amazonaws.com",
    "eventName": "ReserveContact",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "Coral/Jakarta",
    "requestParameters": {
        "satelliteArn":
 "arn:aws:groundstation::123456789012:satellite/11111111-2222-3333-4444-555555555555",
        "groundStation": "Ohio 1",
        "startTime": 1558356107,
        "missionProfileArn": "arn:aws:groundstation:us-east-2:123456789012:mission-
profile/11111111-2222-3333-4444-555555555555",
        "endTime": 1558356886
    },
    "responseElements": {
        "contactId": "11111111-2222-3333-4444-555555555555"
    },
    "requestID": "11111111-2222-3333-4444-555555555555",
    "eventID": "11111111-2222-3333-4444-555555555555",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "11111111-2222-3333-4444-555555555555"
}
```

Monitoring AWS Ground Station

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Ground Station. AWS provides the following monitoring tools to watch AWS Ground Station, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch Events* delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events enables automated event-driven computing, as you can write rules that watch for certain events and trigger automated actions in other AWS services when these events happen. For more information, see the Amazon CloudWatch Events User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Automating AWS Ground Station with CloudWatch Events

Amazon CloudWatch Events enables you to automate your AWS services and respond automatically to system events such as application availability issues or resource changes. Events from AWS services are delivered to CloudWatch Events in near real time. You can write simple rules to indicate which events are of interest to you, and what automated actions to take when an event matches a rule. The actions that can be automatically triggered include the following:

- Invoking an AWS Lambda function
- Invoking Amazon EC2 Run Command
- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine
- Notifying an Amazon SNS topic or an AWS SMS queue

Some examples of using CloudWatch Events with AWS Ground Station include:

- Invoking a Lambda function to start or wake an Amazon EC2 instance during a contact's pre-pass state.
- Notifying an Amazon SNS topic whenever a contact is completed.

For more information, see the Amazon CloudWatch Events User Guide.

Example CloudWatch Events

Ground Station Contact Stage Change

```
{
    "version": "0",
    "id": "01234567-0123-0123",
    "detail-type": "Ground Station Contact State Change",
    "source": "aws.groundstation",
    "account": "123456789012",
```

```
    "time": "2019-05-30T17:40:30Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:groundstation:us-
west-2:123456789012:contact/11111111-2222-3333-4444-555555555555"
    ],
    "detail": {
        "contactId": "11111111-2222-3333-4444-555555555555",
        "groundstationId": "11111111-2222-3333-4444-555555555555",
        "missionProfileArn": "arn:aws:groundstation:us-west-2:123456789012:mission-
profile/11111111-2222-3333-4444-555555555555",
        "satelliteArn":
 "arn:aws:groundstation::123456789012:satellite/11111111-2222-3333-4444-555555555555",
        "contactStatus": "PREPASS"
    }
}
```

Document History for the AWS Ground Station User Guide

The following table describes the important changes to the documentation since the last release of AWS Ground Station.

update-history-change	update-history-description	update-history-date
New Getting Started topic (p. 40)	Updated the Getting Started topic, which includes the most current AWS CloudFormation templates.	July 1, 2019
Kindle version (p. 40)	Published Kindle version of the *AWS Ground Station User Guide*.	June 20, 2019
New service and guide (p. 40)	This is the initial release of AWS Ground Station and the *AWS Ground Station User Guide*.	May 23, 2019

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.