User Guide

AWS Ground Station

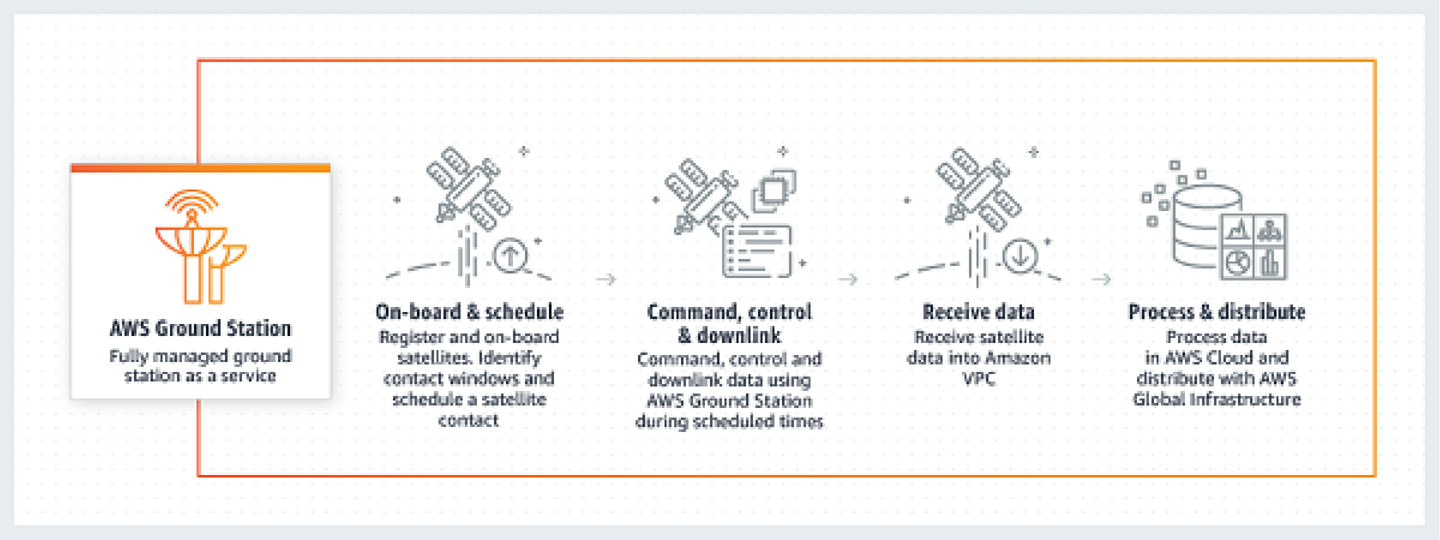
AWS Ground Station: User Guide

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.


AWS Ground Station
Fully managed ground station as a service
On-board & schedule
Register and on-board satellites. Identify contact windows and schedule a satellite contact
Command, control & downlink
Command, control and downlink data using AWS Ground Station during scheduled times
Receive data
Receive satellite data into Amazon VPC
Process & distribute
Process data in AWS Cloud and distribute with AWS Global Infrastructure

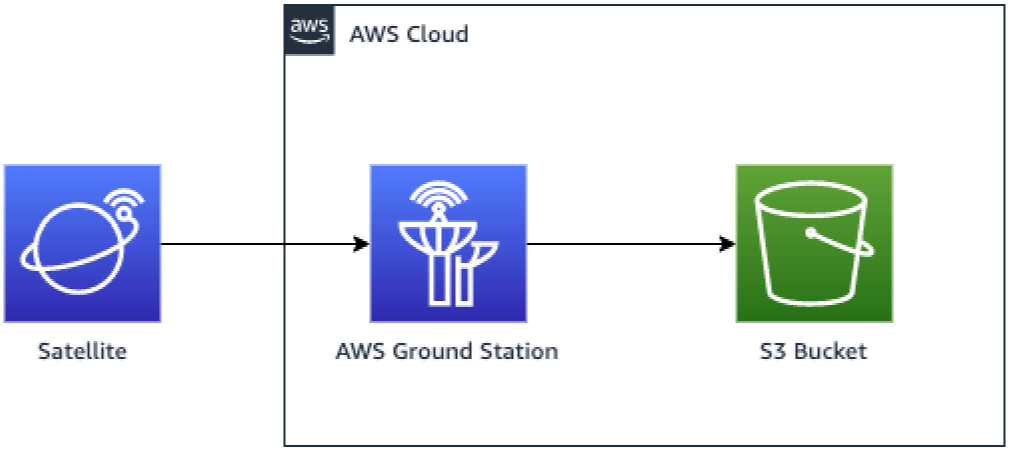
How AWS Ground Station Works

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 or delivered asynchronously to an Amazon Simple Storage Service (Amazon S3) bucket in your account.

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.

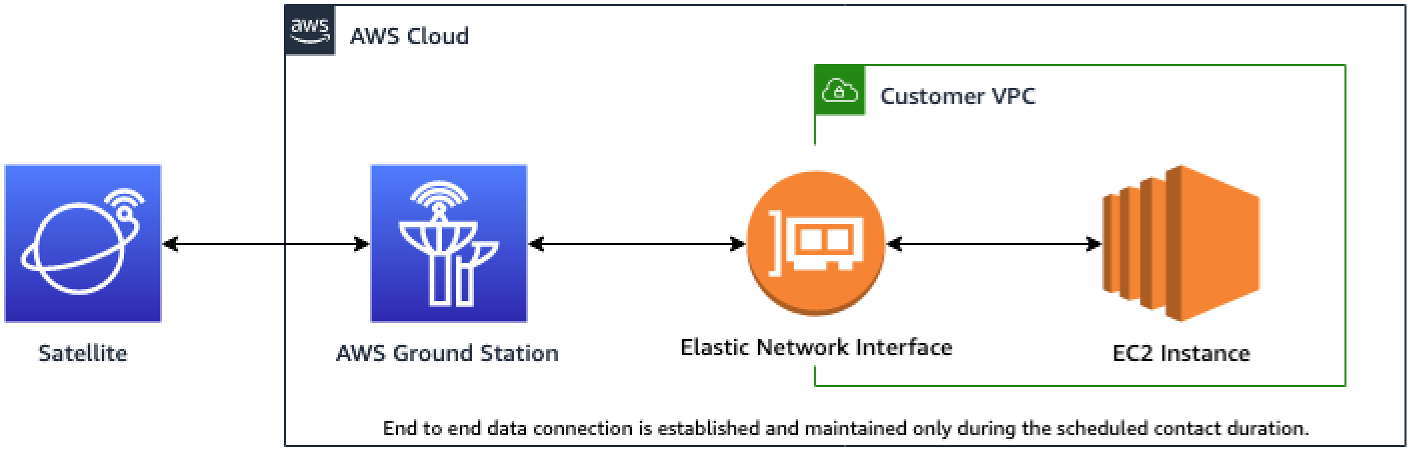
Data Delivery to Amazon S3

With data delivery to Amazon S3, your contact data is delivered asynchronously to an Amazon S3 bucket in your account. Your contact data is delivered as packet capture (pcap) files to allow replaying the contact data into a Software Defined Radio (SDR) or to extract the payload data from the pcap files for processing. The pcap files are delivered to your Amazon S3 bucket every 30 seconds as contact data is received by the antenna hardware to allow processing contact data during the contact if desired. Once received, you can process the data using your own post-processing software or use other AWS services like Amazon SageMaker or Amazon Rekognition. Data delivery to Amazon S3 is only available for downlinking data from your satellite; it is not possible to uplink data to your satellite from Amazon S3.



Data Delivery to Amazon EC2

With data delivery to Amazon EC2, your contact data is streamed to and from your Amazon EC2 instance. You can process your data in real-time on your Amazon EC2 instance or forward the data for post-processing.


AWS Cloud
Customer VPC
Satellite
AWS Ground Station
Elastic Network Interface
EC2 Instance
End to end data connection is established and maintained only during the scheduled contact duration.

More Information

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

- You can receive *narrowband* RF data in S-band (2200 to 2300 MHz) or X-band (7750 to 8400 MHz) at bandwidths up to 54 MHz.
  - S-Band RF data is digitized and provided as a digital stream in VITA-49 Signal Data/IP format.
  - X-Band intermediate frequency (IF) data is digitized and provided as a digital stream in VITA-49 Signal Data/IP format.
- You can receive *wideband* demodulated/decoded data in X-band (7750 to 8400 MHz) at bandwidths up to 500 MHz
  - X-Band intermediate frequency (IF) data is demodulated, decoded, and provided as a digital stream in VITA-49 Extension Data/IP format.
- You can receive *wideband* Digital Intermediate Frequency (DigIF) data from 40 MHz to 400 MHz of bandwidth via the AWS Ground Station Agent.
  - See AWS Ground Station Agent User Guide for more information about the AWS Ground Station Agent and Wideband DigIF Data Delivery.
- You can transmit RF data in S-Band (2025 to 2120 MHz) at bandwidths up to 54 MHz.

  - The RF data is provided to AWS Ground Station as a digital stream in VITA-49 Signal Data/IP format.
- You must run AWS Ground Station from an AWS Region that supports AWS Ground Station. To see a list of supported regions, see the global infrastructure Region Table.
- You can deliver data to an Amazon EC2 instance running in the same region as the antenna, or you can use cross-region data delivery to send your data from an antenna to an Amazon EC2 instance in your preferred AWS Region. The following antenna-to-destination regions are currently available:
  - US East (Ohio) Region (us-east-2) to US West (Oregon) Region (us-west-2)
  - US West (Oregon) Region (us-west-2) to US East (Ohio) Region (us-east-2)

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.

Core Components

Dataflow endpoint groups, configs, and mission profiles are core components of AWS Ground Station. These components determine how you schedule your contacts, how the antennas communicate with your satellites, and where your data is delivered. Before getting started with AWS Ground Station, we recommend that you learn about these components. Examples are provided in their respective sections.

Topics

- Dataflow Endpoint Groups

- Configs
- Mission Profiles

Dataflow Endpoint Groups

Dataflow endpoints define the location where you want the data to be streamed to or from during contacts. The endpoints are identified by a name of your choosing when executing contacts. These names do not need to be unique. This allows multiple contacts to be executed at the same time using the same mission profile.

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) of a role that AWS Ground Station will assume to create Elastic Network Interfaces (ENIs) in your VPC. These ENIs serve as the ingress and egress points of data streamed during a contact.
- `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.

The IAM role passed into `roleArn` must have a trust policy that allows the `groundstation.amazonaws.com` service principal to assume the role. See the Example Trust Policy section below for an example. During endpoint creation the endpoint resource id does not exist, so the trust policy must use an asterisk (`*`) in place of *`your-endpoint-id`*. This can be updated after creation to use the endpoint resource id in order to scope the trust policy to that specific dataflow endpoint group.

The IAM role must have an IAM policy that allows AWS Ground Station to set up the ENIs. See the Example Role Policy section below for an example.

Example Trust Policy

For more information on how to update a role's trust policy, see Managing IAM roles in the IAM User Guide.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "groundstation.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "your-account-id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:groundstation:dataflow-endpoint-region:your-
account-id:dataflow-endpoint-group/your-endpoint-id"
        }
      }
    }
  ]
}
```

Example Role Policy

For more information on how to update or attach a role policy, see Managing IAM policies in the IAM User Guide.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DeleteNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeSecurityGroups"
      ]
    }
  ]
}
```

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 concurrently, but those contacts must be executed on different dataflow endpoint groups.

Dataflow endpoint groups must be in a `HEALTHY` state to schedule contacts using them. Listed below are the reasons your dataflow endpoint groups may not be in a `HEALTHY` state as well as the appropriate corrective action to take.

- `NO_REGISTERED_AGENT` - Start your EC2 instance, which will register the agent. Note that you must have a valid controller config file for this call to be successful. Refer to the AWS Ground Station Agent User Guide for details on configuring that file.
- `INVALID_IP_OWNERSHIP` - Use the DeleteDataflowEndpointGroup API to delete the Dataflow Endpoint Group, then use the CreateDataflowEndpointGroup API to recreate the Dataflow Endpoint Group using IP addresses and ports that are associated with the EC2 instance.
- `UNVERIFIED_IP_OWNERSHIP` - IP address has not been validated yet. Validation occurs periodically so this should resolve itself.
- `NOT_AUTHORIZED_TO_CREATE_SLR` - Account is not authorized to create the necessary Service-Linked Role. Check the troubleshooting steps in Using service-linked roles for Ground Station

See the following documentation for more information about how to perform operations on dataflow endpoint groups using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::DataflowEndpointGroup CloudFormation resource type
- Dataflow Endpoint Group AWS CLI reference
- Dataflow Endpoint Group API reference

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.

See the following documentation for more information about how to perform operations on configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API. Links to documentation for specific config types are also provided below.

- AWS::GroundStation::Config CloudFormation resource type
- Config AWS CLI reference
- Config API reference

Dataflow Endpoint Config

> **Note**
>
> Dataflow endpoint configs are only used for data delivery to Amazon EC2 and are not used for data delivery to Amazon S3.

You can use dataflow endpoint configs to specify which dataflow endpoint in a dataflow endpoint group from which or to which you want data to flow during a contact. The two parameters of a dataflow endpoint config specify the name and region of the dataflow endpoint. When reserving a contact, AWS Ground Station analyzes the mission profile you specified and attempts to find a dataflow endpoint *group* that contains all of the dataflow *endpoints* specified by the dataflow endpoint *configs* contained in your mission profile.

The `dataflowEndpointName` property of a dataflow endpoint config specifies which dataflow endpoint in a dataflow endpoint group to which or from which data will flow during a contact.

The `dataflowEndpointRegion` property specifies which region the dataflow endpoint resides in. If a region is specified in your dataflow endpoint config, AWS Ground Station looks for a dataflow endpoint in the region specified. If no region is specified, AWS Ground Station will default to the contact's ground station region. A contact is considered a cross region data delivery contact if your dataflow endpoint's region is not the same as the contact's ground station region.

See the following documentation for more information about how to perform operations on dataflow endpoint configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config DataflowEndpointConfig CloudFormation property
- Config AWS CLI reference (see the `dataflowEndpointConfig -> (structure)` section)
- DataflowEndpointConfig API reference

S3 Recording Config

> **Note**
>
> S3 recording configs are only used for data delivery to Amazon S3 and are not used for data delivery to Amazon EC2.

You can use S3 recording configs to specify an Amazon S3 bucket to which you want downlinked data delivered. The two parameters of an S3 recording config specify the Amazon S3 bucket and IAM role for AWS Ground Station to assume when delivering the data to your Amazon S3 bucket. The IAM role and Amazon S3 bucket specified must meet the following criteria:

- The Amazon S3 bucket's name must begin with `aws-groundstation`.
- The IAM role must have a trust policy that allows the `groundstation.amazonaws.com` service principal to assume the role. See the Example Trust Policy section below for an example. During config creation the config resource id does not exist, the trust policy must use an asterisk (`*`) in place of *`your-config-id`* and can be updated after creation with the config resource id.
- The IAM role must have an IAM policy that allows the role to perform the `s3:GetBucketLocation` action on the bucket and `s3:PutObject` action on the bucket's

objects. If the Amazon S3 bucket has a bucket policy, then the bucket policy must also allow the IAM role to perform these actions. See the Example Role Policy section below for an example.

Example Trust Policy

For more information on how to update a role's trust policy, see Managing IAM roles in the IAM User Guide.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "groundstation.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "your-account-id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:groundstation:config-region:your-account-id:config/
s3-recording/your-config-id"
        }
      }
    }
  ]
}
```

Example Role Policy

For more information on how to update or attach a role policy, see Managing IAM policies in the IAM User Guide.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::your-bucket-name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::your-bucket-name/*"
      ]
    }
  ]
}
```

See the following documentation for more information about how to perform operations on S3 recording configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config S3RecordingConfig CloudFormation property
- Config AWS CLI reference (see the `s3RecordingConfig -> (structure)` section)
- S3RecordingConfig API reference

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.

See the following documentation for more information about how to perform operations on tracking configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config TrackingConfig CloudFormation property
- Config AWS CLI reference (see the `trackingConfig -> (structure)` section)
- TrackingConfig API reference

Antenna Downlink Config

You can use antenna downlink configs to configure the antenna for downlink during your contact. They consist of a spectrum config that specifies the frequency, bandwidth, and polarization that should be used during your downlink contact. If your downlink use case requires demodulation or decoding, see the Antenna Downlink Demod Decode Config.

See the following documentation for more information about how to perform operations on antenna downlink configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config AntennaDownlinkConfig CloudFormation property
- Config AWS CLI reference (see the `antennaDownlinkConfig -> (structure)` section)
- AntennaDownlinkConfig API reference

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.

See the following documentation for more information about how to perform operations on antenna downlink demod decode configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config AntennaDownlinkDemodDecodeConfig CloudFormation property
- Config AWS CLI reference (see the `antennaDownlinkDemodDecodeConfig -> (structure)` section)

- AntennaDownlinkDemodDecodeConfig API reference

Antenna Uplink Config

You can use antenna uplink configs to configure the antenna for uplink during your contact. They consist of a spectrum config with frequency, polarization, and target effective isotropic radiated power (EIRP). For information about how to configure a contact for uplink loopback, see Uplink Echo Config.

See the following documentation for more information about how to perform operations on antenna uplink configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config AntennaUplinkConfig CloudFormation property
- Config AWS CLI reference (see the `antennaUplinkConfig -> (structure)` section)
- AntennaUplinkConfig API reference

Uplink Echo Config

Uplink echo configs tell the antenna how to execute an uplink echo. This echoes the signal sent by the antenna back to your dataflow endpoint. An 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 an uplink echo.

See the following documentation for more information about how to perform operations on uplink echo configs using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

- AWS::GroundStation::Config UplinkEchoConfig CloudFormation property
- Config AWS CLI reference (see the `uplinkEchoConfig -> (structure)` section)
- UplinkEchoConfig API reference

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 how the antenna will be configured and where data will go during your contact.

Aside from tracking configs, all configs are contained in the `dataflowEdges` field of the mission profile. A single dataflow 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 AWS Ground Station from where and to where 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.

See the following documentation for more information about how to perform operations on mission profiles using AWS CloudFormation, the AWS Command Line Interface, or the AWS Ground Station API.

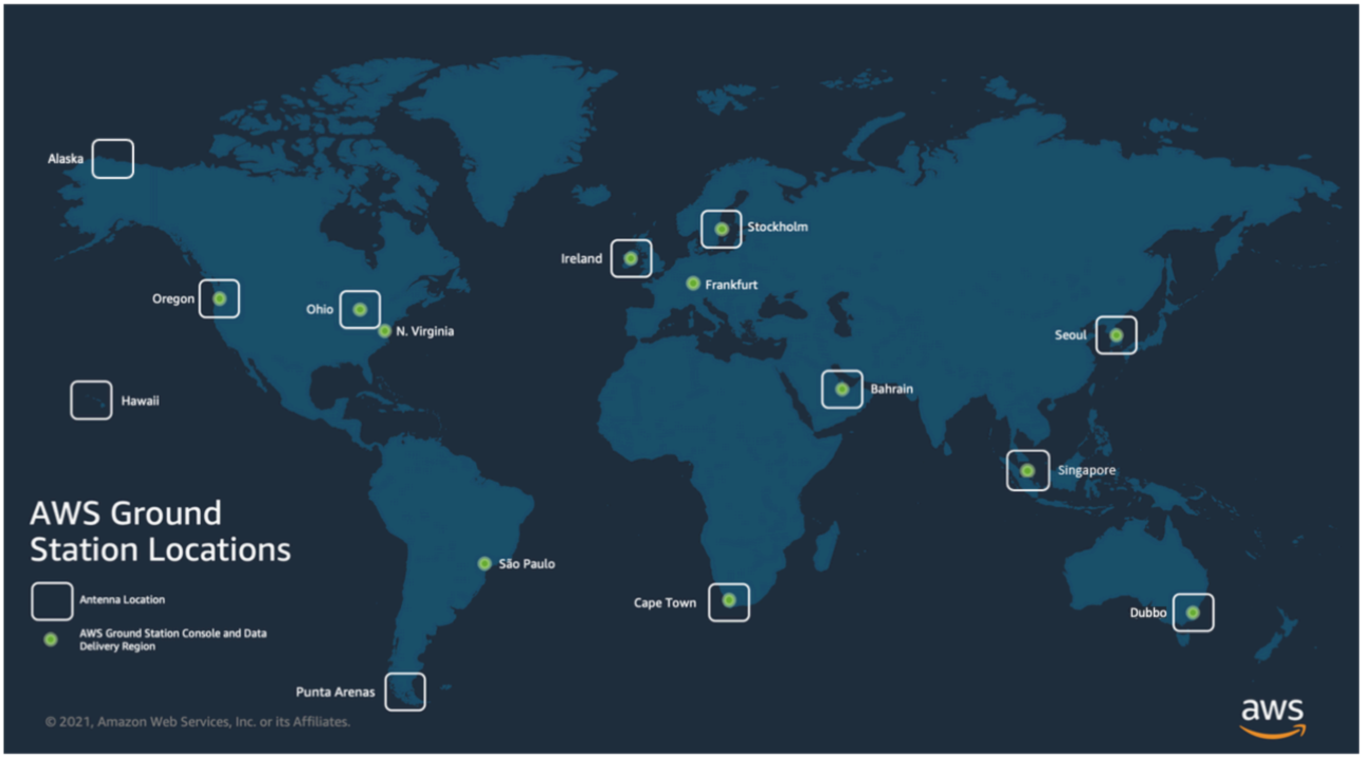
- AWS::GroundStation::MissionProfile CloudFormation resource type
- Mission Profile AWS CLI reference
- Mission Profile API reference

AWS Ground Station Locations

Customers can transmit and receive data using AWS Ground Station antennas in the following locations: US (Oregon), US (Ohio), US (Alaska), Middle East (Bahrain), Europe (Stockholm), Asia Pacific (Dubbo), Europe (Ireland), Africa (Cape Town), US (Hawaii), Asia Pacific (Seoul), Asia Pacific (Singapore), and South America (Punta Arenas).

Customers can deliver data and configure their contacts with the AWS Ground Station console in the following regions: US West (Oregon), US East (Ohio), Middle East (Bahrain), Europe (Stockholm), Asia Pacific (Dubbo), Europe (Ireland), Africa (Cape Town), US East (N. Virginia), Europe (Frankfurt), Asia Pacific (Seoul), Asia Pacific (Singapore), and South America (São Paulo).

Note: You can only create AWS Ground Station resources in the regions that host the AWS Ground Station console mentioned in the prior paragraph.



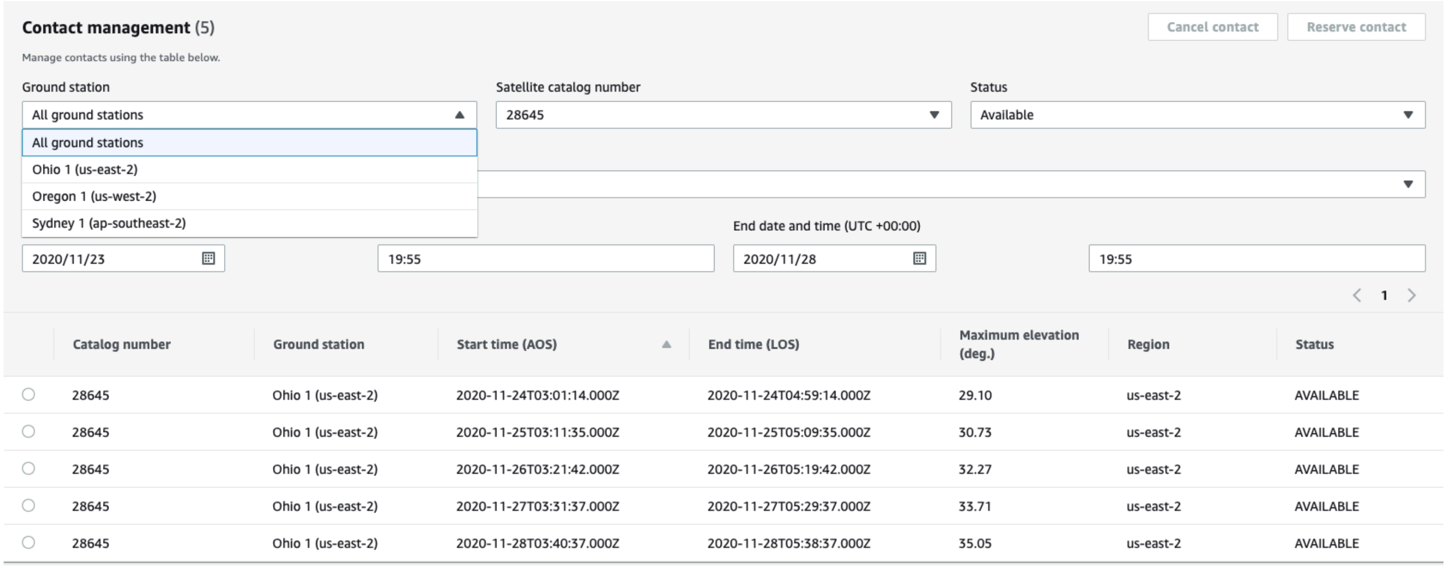
Topics

- Finding the AWS Region for a Ground Station

Finding the AWS Region for a Ground Station

The AWS Global Network includes Ground Station locations that are not physically located in the AWS Region to which they are connected. Listing and reserving contacts at one of these Ground Station locations must be performed using the AWS Region to which the Ground Station is connected.

There are multiple methods of determining a Ground Station's AWS Region. The AWS Ground Station console page displays the Ground Station's AWS Region when displaying it in both the filters and contacts table as shown in the image below. The AWS SDK contains the Ground Station's AWS Region in the ListGroundStation response. Finally, the AWS CLI contains the Ground Station's AWS Region in the list-ground-stations response.



Contact management (5)

Manage contacts using the table below.

Ground station: All ground stations

- All ground stations
- Ohio 1 (us-east-2)
- Oregon 1 (us-west-2)
- Sydney 1 (ap-southeast-2)

Satellite catalog number: 28645

Status: Available

2020/11/23 19:55

End date and time (UTC +00:00): 2020/11/28 19:55

	Catalog number	Ground station	Start time (AOS)	End time (LOS)	Maximum elevation (deg.)	Region	Status
○	28645	Ohio 1 (us-east-2)	2020-11-24T03:01:14.000Z	2020-11-24T04:59:14.000Z	29.10	us-east-2	AVAILABLE
○	28645	Ohio 1 (us-east-2)	2020-11-25T03:11:35.000Z	2020-11-25T05:09:35.000Z	30.73	us-east-2	AVAILABLE
○	28645	Ohio 1 (us-east-2)	2020-11-26T03:21:42.000Z	2020-11-26T05:19:42.000Z	32.27	us-east-2	AVAILABLE
○	28645	Ohio 1 (us-east-2)	2020-11-27T03:31:37.000Z	2020-11-27T05:29:37.000Z	33.71	us-east-2	AVAILABLE
○	28645	Ohio 1 (us-east-2)	2020-11-28T03:40:37.000Z	2020-11-28T05:38:37.000Z	35.05	us-east-2	AVAILABLE

Topics

- Example Ground Station Located Outside of an AWS Region

Example Ground Station Located Outside of an AWS Region

Hawaii 1 is an example of a Ground Station location that is not physically located in the AWS Region to which it is connected. The Hawaii 1 Ground Station is located in Hawaii, USA but is connected to the us-west-2 (Oregon) AWS Region. In order to list and reserve contacts using Hawaii 1, you must have a mission profile configured in the us-west-2 (Oregon) AWS Region and use the us-west-2 (Oregon) AWS Region in the AWS Ground Station console, AWS CLI, or AWS SDK.

- To list and reserve contacts for Hawaii 1 in the AWS Ground Station console you must use the AWS Ground Station console in the us-west-2 (Oregon) region.
- To list and reserve contacts for Hawaii 1 using the AWS CLI you must specify the region as us-west-2 using the `--region` CLI argument.
- To list and reserve contacts for Hawaii 1 using the AWS SDK you must set the region of your client to us-west-2. How you set this depends on the programming language your are using. An example of how to set this using JavaScript is described in the AWS SDK for JavaScript documentation. For more information, refer to the language specific SDK documentation.

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. Use the following steps to set up your account.

Topics

Sign up for an AWS account

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

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

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

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

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

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

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

Sign in as the user with administrative access

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

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

Assign access to additional users

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

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

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

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

Add Ground Station Permissions to Your AWS Account

To use AWS Ground Station without requiring an administrative user, you need to create a new policy and attach it to your AWS account.

1. Sign in to the AWS Management Console and open the IAM console.
2. Create a new policy. Use the following steps:

   a. In the navigation pane, choose **Policies** and then choose **Create Policy**.

   b. In the **JSON** tab, edit the JSON with one of the following values. Use the JSON that works best for your application.

      - For Ground Station administrative privileges, set **Action** to **groundstation:*** as follows:

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

- For Read-only privileges, 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
                }
            }
        }
    ]
}
```

3. In the IAM console, attach the policy you created to the desired user.

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

Customer Onboarding

To complete registration for your AWS Ground Station account, see the Satellites and Resources section in the AWS Ground Station console page for onboarding details. The AWS Ground Station team will work with you to onboard your satellites to the service. Once you onboard your satellite, the satellite will be available to use when managing a contact. Instructions for managing a contact are provided in Listing and Reserving Contacts.

Onboarding your satellite(s) will grant you access to send and receive data to and from the satellite. In addition to onboarding your own satellites, customers may also onboard the following satellites to downlink direct broadcast data using AWS Ground Station:

- Aqua
- SNPP
- JPSS-1/NOAA-20
- Terra

Once onboarded, these satellites can be accessed for immediate use. AWS Ground Station maintains a number of preconfigured AWS CloudFormation templates to make getting started with the service easier. Instructions and details for accessing and using this template are provided in the Create Your Resources Using a AWS CloudFormation Template section of the user guide.

For more information about these satellites and the kind of data they transmit, see Aqua, JPSS-1/NOAA-20 and SNPP, and Terra.

Next Steps

Your AWS Ground Station account is now set up and ready for configuration. Continue to *Getting Started* to configure your resources to use AWS Ground Station.

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. AWS Ground Station delivers your contact data asynchronously to an Amazon Simple Storage Service (Amazon S3) bucket in your account or synchronously by streaming it to and from an Amazon Elastic Compute Cloud (Amazon EC2) instance in your account. The following steps describe how to configure the resources required to receive contact data asynchronously in an Amazon S3 bucket. See the Data Delivery to Amazon EC2 guide for information about how to use data delivery to Amazon EC2.

Topics

Basic Concepts

Before you begin, you should familiarize yourself with the basic concepts in AWS Ground Station. For more information, see Core Components.

Then, continue on to Prerequisites to learn about prerequisites to getting started with AWS Ground Station.

Prerequisites

Before getting started with AWS Ground Station, ensure you have an AWS account with the proper credentials. Follow the steps in Setting Up AWS Ground Station.

Note

If you will be using Wideband DigIF Data Delivery, see the AWS Ground Station Agent User Guide for instructions.

Otherwise, continue on to Step 1: Choose an AWS CloudFormation Template.

Step 1: Choose an AWS CloudFormation Template

After you onboard your satellite, you need to define mission profiles to define the AWS Ground Station antenna configuration to downlink data from your satellite. To assist you with this process, we provide preconfigured AWS CloudFormation templates for both Narrowband and Wideband DigIF Data Delivery that use public broadcast satellites. These templates make it easy for you to start using AWS Ground Station. For more information about AWS CloudFormation, see What is AWS CloudFormation?

Depending on the type of contact you'd like to take, pick the appropriate CFN template type from the list below:

- Narrowband S3 Data Delivery AWS CloudFormation Templates.
- Wideband DigIf S3 Data Delivery AWS CloudFormation Templates.

If you don't want to use one of the premade AWS CloudFormation templates, you can view instructions on Building your own template.

Narrowband S3 Data Delivery AWS CloudFormation Templates

Preconfigured Templates

Today, you can configure multiple streams of data per contact to flow into an S3 bucket. These data streams are available in two different formats. Data streams containing VITA-49 Signal/ IP data can be configured for S-Band and X-Band signals up to 54 MHz in bandwidth. VITA-49 Extension data/IPs can be configured for demodulated and/or decoded X-Band signals up to 500 MHz in bandwidth.

AWS Ground Station provides templates for both data stream formats that demonstrate how to use the service. Use this guide to find the right template for you.

Available templates

You can use a preconfigured template to receive direct broadcast data from the Aqua, SNPP, JPSS-1/NOAA-20, and Terra satellites. These AWS CloudFormation templates contain the required AWS Ground Station and Amazon S3 resources to schedule and execute contacts and receive the data in an Amazon S3 bucket in your account. If Aqua, SNPP, JPSS-1/NOAA-20, and Terra are not onboarded to your account, see Customer Onboarding.

Narrowband Data Delivery Templates

If you are using narrowband data delivery for your contact, use the AWS CloudFormation templates below.

- The AWS CloudFormation template named `AquaSnppJpss-1DemodDecodeS3DataDelivery.yml` contains an Amazon S3 bucket and the required AWS Ground Station resources to schedule contacts and receive demodulated and decoded direct broadcast data. This template is a good starting point if you plan to process the data using NASA Direct Readout Labs software (RT-STPS and IPOPP).

  To download the template using AWS CLI, use the following command:

  ```
  aws s3 cp s3://groundstation-cloudformation-templates-us-west-2/
  AquaSnppJpss-1DemodDecodeS3DataDelivery.yml .
  ```

  You can view and download the template in the console by navigating to the following URL in your browser:

  ```
  https://s3.console.aws.amazon.com/s3/object/groundstation-cloudformation-templates-
  us-west-2/AquaSnppJpss-1DemodDecodeS3DataDelivery.yml
  ```

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

  ```
  https://groundstation-cloudformation-templates-us-west-2.s3.us-west-2.amazonaws.com/
  AquaSnppJpss-1DemodDecodeS3DataDelivery.yml
  ```

- The AWS CloudFormation template named `AquaSnppJpss-1TerraDigIfS3DataDelivery.yml` contains an Amazon S3 bucket and the required AWS Ground Station resources to schedule contacts and receive VITA-49 Signal/IP direct broadcast data. This template is a good starting point if you plan to process the data using a software defined radio (SDR) to demodulate and decode the data before post-processing.

  To download the template using AWS CLI, use the following command:

```
aws s3 cp s3://groundstation-cloudformation-templates-us-west-2/
AquaSnppJpss-1TerraDigIfS3DataDelivery.yml .
```

You can view and download the template in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-cloudformation-templates-
us-west-2/AquaSnppJpss-1TerraDigIfS3DataDelivery.yml
```

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

```
https://groundstation-cloudformation-templates-us-west-2.s3.us-west-2.amazonaws.com/
AquaSnppJpss-1TerraDigIfS3DataDelivery.yml
```

What resources do these template define?

Both of the templates contain the same resources, with the sole difference being the antenna configs. See the **Antenna Config** description below for more information.

- **Amazon S3 Bucket** - The bucket to which the downlinked data will be delivered. The name of this bucket starts with `aws-groundstation` to meet criteria described in S3 Recording Config.
- **IAM Role** - A role assumable by the `groundstation.amazonaws.com` service principal that AWS Ground Station assumes when writing the downlinked data to your Amazon S3 bucket.
- **Amazon S3 Bucket Policy** - A policy that allows the IAM Role to perform the following actions on your Amazon S3 bucket and its objects:
  - `s3:GetBucketLocation`
  - `s3:PutObject`
- **Tracking Config** - An AWS Ground Station tracking config that defines how the antenna system tracks your satellite as it moves through the sky.
- **S3 Recording Config** - An AWS Ground Station S3 recording config that references the Amazon S3 bucket and IAM role for AWS Ground Station to use when delivering your data.
- **Antenna Config** - An AWS Ground Station antenna config that specifies how to configure the AWS Ground Station antenna during a contact. The `AquaSnppJpss-1DemodDecodeS3DataDelivery.yml` template contains an antenna downlink demod decode config that configures the AWS Ground Station antenna to demodulate

and decode the downlinked data before delivering it to your Amazon S3 bucket. The `AquaSnppJpss-1TerraDigIfS3DataDelivery.yml` instead contains an antenna downlink config that configures the AWS Ground Station antenna to deliver the data to your Amazon S3 as VITA-49 Signal/IP packets.

- **Mission Profile** - An AWS Ground Station mission profile that groups all of the AWS Ground Station configs together to allow you to schedule and execute contacts using the configurations referenced.

Wideband DigIf S3 Data Delivery AWS CloudFormation Templates

Wideband DigIF Data Delivery Templates

If you are using Wideband Digital Intermediate Frequency (DigIF) data delivery for your contact, use the AWS CloudFormation templates below.

- The AWS CloudFormation template named `DirectBroadcastSatelliteWbDigIfS3DataDelivery.yml` contains an Amazon S3 bucket and the required AWS Ground Station resources to schedule contacts and receive VITA-49 Signal/IP direct broadcast data via the AWS Ground Station Agent. This template is a good starting point if you plan to process the data using a software defined radio (SDR) to demodulate and decode the data before post-processing. For more information about the AWS Ground Station Agent, see AWS Ground Station Agent User Guide.

  To download the template using AWS CLI, use the following command:

  ```
  aws s3 cp s3://groundstation-cloudformation-templates-us-west-2/agent/s3_recording/
  DirectBroadcastSatelliteWbDigIfS3DataDelivery.yml .
  ```

  You can view and download the template in the console by navigating to the following URL in your browser:

  ```
  https://s3.console.aws.amazon.com/s3/object/groundstation-cloudformation-templates-
  us-west-2/agent/s3_recording/DirectBroadcastSatelliteWbDigIfS3DataDelivery.yml
  ```

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

```
https://groundstation-cloudformation-templates-us-west-2.s3.us-west-2.amazonaws.com/
agent/s3_recording/DirectBroadcastSatelliteWbDigIfS3DataDelivery.yml
```

What resources does this template define?

- **Amazon S3 Bucket** - The bucket to which the downlinked data will be delivered. The name of this bucket starts with `aws-groundstation` to meet criteria described in S3 Recording Config.
- **IAM Role** - A role assumable by the `groundstation.amazonaws.com` service principal that AWS Ground Station assumes when writing the downlinked data to your Amazon S3 bucket.
- **Amazon S3 Bucket Policy** - A policy that allows the IAM Role to perform the following actions on your Amazon S3 bucket and its objects:
  - `s3:GetBucketLocation`
  - `s3:PutObject`
- **AWS KMS Key** - A AWS KMS Key used to encrypt dataflows.
- **Ground Station Key Role** - The IAM role that AWS Ground Station will assume to access and use the AWS KMS Key to decrypt dataflows
- **Ground Station Key Access Policy** - The IAM policy defining the actions AWS Ground Station can take on the Data Delivery Key
- **Tracking Config** - An AWS Ground Station tracking config that defines how the antenna system tracks your satellite as it moves through the sky.
- **S3 Recording Config** - An AWS Ground Station S3 recording config that references the Amazon S3 bucket and IAM role for AWS Ground Station to use when delivering your data.
- **Antenna Configs for Aqua, SNPP, JPSS-1/NOAA-20, and Terra** - Three separate AWS Ground Station antenna configs that specify how to configure the AWS Ground Station antenna during a contact with Aqua, SNPP, JPSS-1/NOAA-20, and Terra. The template contains an antenna downlink config that configures the AWS Ground Station antenna to deliver the data to your Amazon S3 as VITA-49 Signal/IP packets.
- **Mission Profiles for Aqua, SNPP, JPSS-1/NOAA-20, and Terra** - Three separate AWS Ground Station mission profiles that groups all of the AWS Ground Station configs together to allow you to schedule and execute contacts using the configurations referenced with Aqua, SNPP, JPSS-1/NOAA-20, and Terra.

Building your own template

Configuring the resources to schedule and execute contacts for your own satellites requires you configure the AWS Ground Station resources in your account to match your satellite's settings. This is difficult to do on your own. The AWS Ground Station team is available to help you configure the AWS Ground Station resources in your account to downlink from and uplink to your satellite. To configure your own satellite to use with AWS Ground Station, contact AWS Support.

Step 2: Configure an AWS CloudFormation Stack

After choosing the template that best applies to your use case, configure an AWS CloudFormation stack. The resources that are created in this procedure are configured to the region that you are in when you create them.

1. In the AWS Management Console, choose **Services > CloudFormation**.
2. In the navigation pane, choose **Stacks**. Then, choose **Create stack > With new resources (standard)**.
3. In the **Create Stack** page, specify the template that you selected in the section called "Step 1: Choose an AWS CloudFormation Template" by doing one of the following.

    a. Select **Amazon S3 URL** as your template source, and copy and paste the URL of the template you want to use in **Amazon S3 URL**. Then, choose **Next**.

    b. Select **Upload a template file** as your template source and choose **Choose File**. Upload the template you downloaded in the section called "Step 1: Choose an AWS CloudFormation Template". Then, choose **Next**.
4. Perform the following steps in the **Specify stack details** page:

    a. Enter a name in the **Stack Name** box. We recommend using a simple name to reduce the possibility of errors in the future.

    b. Choose **Next**.
5. Configure stack options and advanced options for your Amazon EC2 instance.

    a. Add any tags and permissions in the **Tags** and **Permissions** sections.

    b. Make any changes for your **Stack policy**, **Rollback configuration**, **Notification options**, and **Stack creation options**.

    c. Choose **Next**.

6. After reviewing your stack details, select the **Capabilities** acknowledgement, and choose **Create stack**.

AWS Ground Station Agent User Guide

Topics

Overview

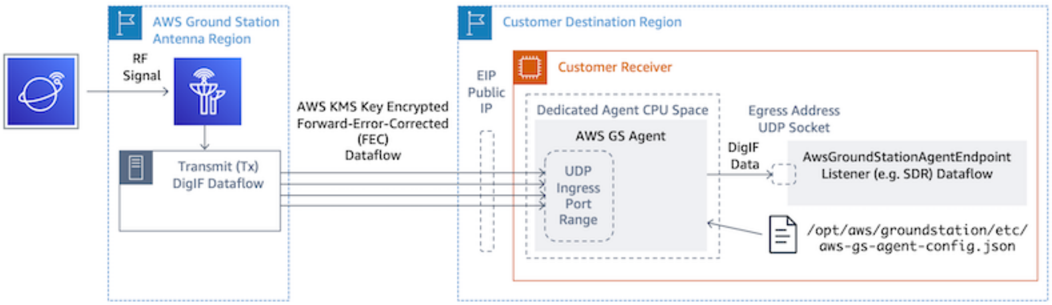
What is the AWS Ground Station Agent?

The AWS Ground Station Agent, available as an RPM, enables AWS Ground Station customers to receive (downlink) synchronous Wideband Digital Intermediate Frequency (DigIF) dataflows during AWS Ground Station contacts. Customers can select two options for data delivery:

1. **Data delivery to an EC2 instance** - Data delivery to a customer owned EC2 instance. AWS Ground Station customers manage the AWS Ground Station Agent. This option may suit you best if you need near real-time data processing. See the Data Delivery to Amazon EC2 guide for information about EC2 data delivery.
2. **Data delivery to an S3 bucket** - Data delivery to a customer owned AWS S3 bucket via a Ground Station managed service. See the Getting Started with AWS Ground Station guide for information about S3 data delivery.

Both modes of data delivery require customers to create a set of AWS resources. The use of CloudFormation templates to create your AWS resources is highly recommended to ensure reliability, accuracy, and supportability. Each contact can only deliver data to EC2 or S3 but not to both simultaneously.

> **Note**
>
> Since S3 data delivery is a Ground Station managed service, this guide focuses on data delivery to your EC2 instance(s).


AWS Ground Station Antenna Region
RF Signal
Transmit (Tx) DigIF Dataflow
AWS KMS Key Encrypted Forward-Error-Corrected (FEC) Dataflow
Customer Destination Region
EIP Public IP
Customer Receiver
Dedicated Agent CPU Space
AWS GS Agent
UDP Ingress Port Range
Egress Address UDP Socket
DigIF Data
AwsGroundStationAgentEndpoint Listener (e.g. SDR) Dataflow
/opt/aws/groundstation/etc/aws-gs-agent-config.json

DigIF dataflow from an AWS Ground Station Antenna Region to your EC2 instance with your Software-Defined Radio (SDR) or similar listener.

Features of the AWS Ground Station Agent

The AWS Ground Station Agent receives Digital Intermediate Frequency (DigIF) downlink data and egresses decrypted data that enables the following:

- DigIF downlink capability from 40 MHz to 400 MHz of bandwidth.
- High rate, low jitter DigIF data delivery to any public IP (AWS Elastic IP) on the AWS network.
- Reliable data delivery using Forward Error Correction (FEC).
- Secure data delivery using a customer managed AWS KMS key for encryption.

Agent Requirements

Note

This AWS Ground Station Agent guide assumes that you have onboarded to Ground Station using the Setting Up AWS Ground Station guide.

The AWS Ground Station Agent receiver EC2 instance requires a set of dependent AWS resources to reliably and securely deliver DigIF data to your endpoints.

1. A VPC in which to launch the EC2 receiver.
2. An AWS KMS Key for data encryption/decryption.
3. An SSH key or EC2 Instance Profile configured for SSM Session Manager.
4. Network/Security Group rules to allow the following:
    1. UDP traffic from AWS Ground Station on the ports specified in your dataflow endpoint group. The agent reserves a range of contiguous ports used to deliver data to the ingress dataflow endpoint(s).
    2. SSH access to your instance (Note: You can alternatively use AWS Session Manager to access your EC2 instance).
    3. Read access to a publicly accessible S3 bucket for agent management.
    4. SSL traffic on port 443 allowing the agent to communicate with the AWS Ground Station service.
    5. Traffic from the AWS Ground Station managed prefix list `com.amazonaws.global.groundstation`.

Additionally, a VPC configuration including a public subnet is required. Refer to the VPC User Guide for background on subnet configuration.

Compatible configurations:

1. An Elastic IP associated with your EC2 instance in a public subnet.
2. An Elastic IP associated with an ENI in a public subnet, attached to your EC2 instance (in any subnet).

You may use the same security group as your EC2 instance or specify one with at least the minimum set of rules consisting of:

- UDP traffic from AWS Ground Station on the ports specified in your dataflow endpoint group.

See the "Wideband DigIF Data Delivery Templates" section of Choose a Template for example AWS CloudFormation EC2 Data Delivery templates with these resources preconfigured.

VPC diagrams

Diagram: An Elastic IP associated with your EC2 instance in a public subnet

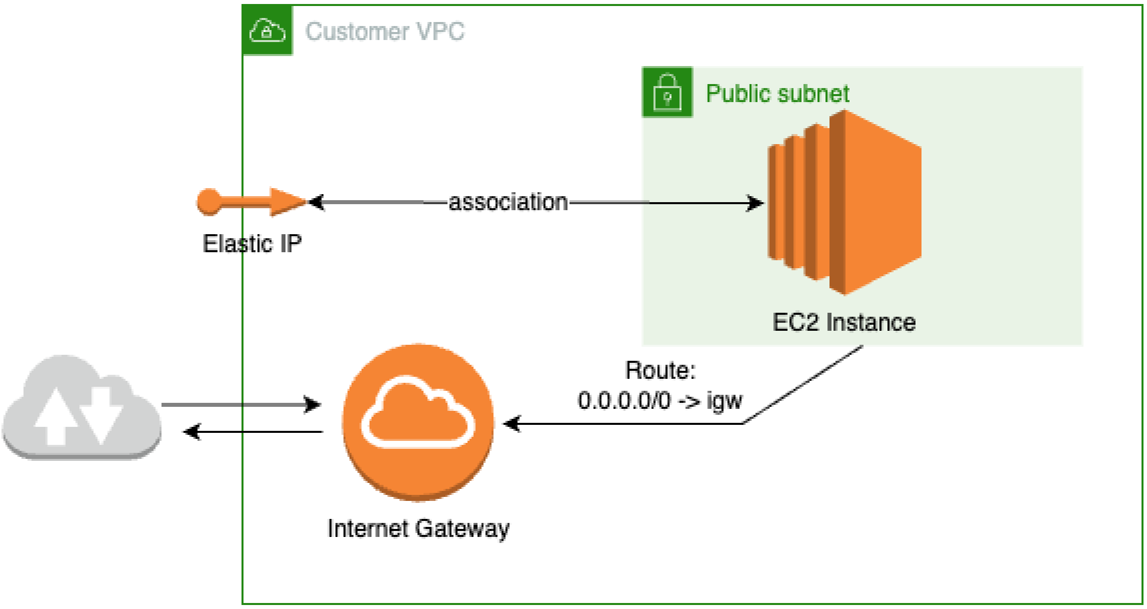
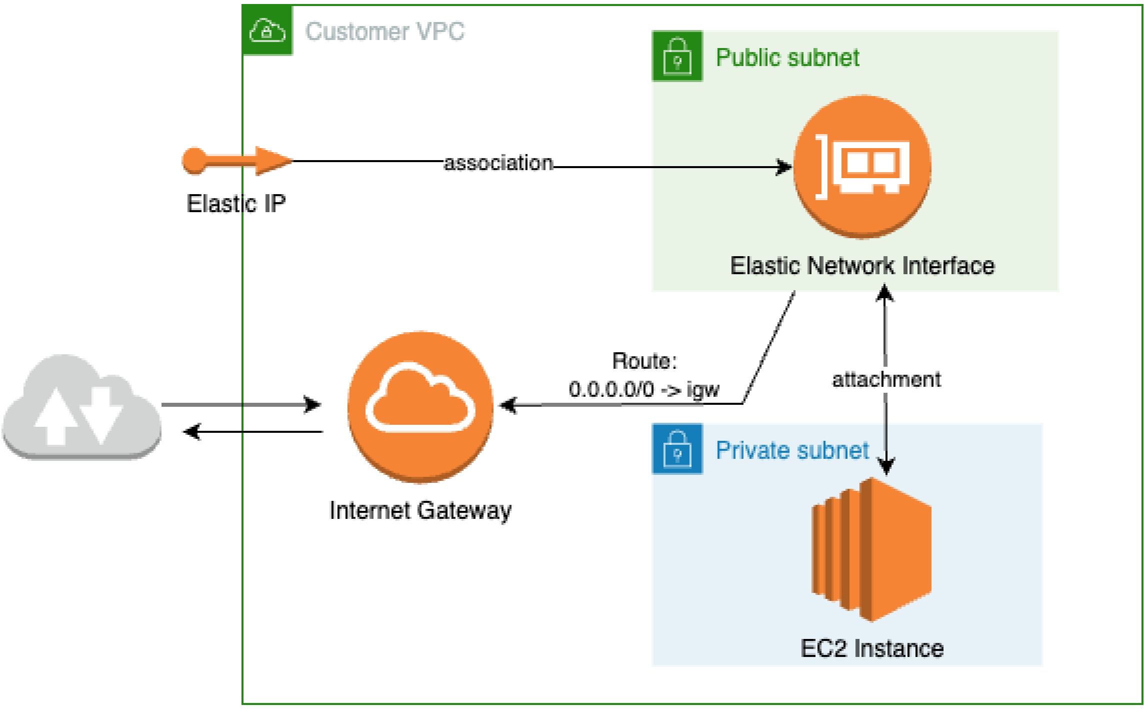


Diagram: An Elastic IP associated with an ENI in a public subnet, attached to your EC2 instance in a private subnet



Supported operating system

Amazon Linux 2 with 5.10+ kernel.

Supported instances types are listed in EC2 Instance Selection and CPU Planning

Data Delivery via AWS Ground Station Agent

The diagrams below provide an overview of how data flows through AWS Ground Station during Wideband Digital Intermediate Frequency (DigIF) contacts.

The AWS Ground Station Agent will handle orchestrating the dataplane components for a contact. Prior to scheduling a contact the agent must be correctly configured, started, and it must be registered (registration is automatic upon agent startup) with AWS Ground Station. In addition, the data receiving software (such as a software defined radio) must be running and configured to receive data at the AwsGroundStationAgentEndpoint egressAddress.

Behind the scenes, the AWS Ground Station Agent will receive tasking from AWS Ground Station and undo the AWS KMS encryption that was applied in transit, before forwarding it to the destination endpoint egressAddress where your Software Defined Radio (SDR) is listening. The AWS Ground Station Agent and its underlying components will respect the CPU boundaries set in the configuration file to ensure it does not impact performance of other applications running on the instance.

Customers must have the AWS Ground Station Agent running on the receiver instance involved in the contact. A single AWS Ground Station Agent is able to orchestrate multiple dataflows, as seen below, if the customer prefers to receive all dataflows on a single receiver instance.

Multiple Dataflows, Single Receiver

Example Scenario:

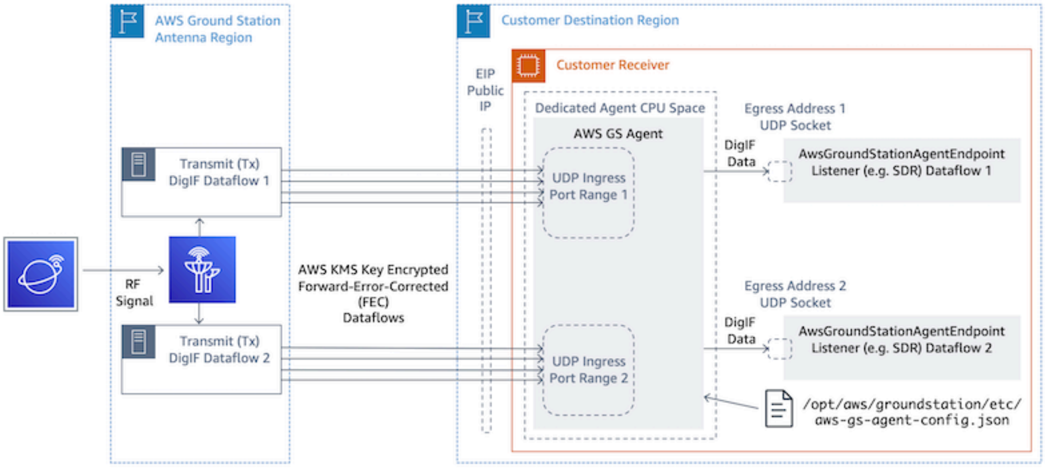
The customer would like to receive two antenna downlinks as DigIF dataflows at the same EC2 receiver instance. The two downlinks will be 200MHz and 100MHz.

AwsGroundStationAgentEndpoints:

There will be two `AwsGroundStationAgentEndpoint` resources, one for each dataflow. Both endpoints will have the same public IP address (`ingressAddress.socketAddress.name`). The ingress `portRange`'s should not overlap, as the dataflows are being received at the same EC2 instance. Both `egressAddress.socketAddress.port`'s must be unique.

CPU Planning:

- 1 core (2 vCPU) for running the single AWS Ground Station Agent on the instance.
- 6 cores (12 vCPU) to receive DigIF Dataflow 1 (200MHz lookup in CPU Core Planning table).
- 4 cores (8 vCPU) to receive DigIF Dataflow 2 (100MHz lookup in CPU Core Planning table).
- Total Dedicated Agent CPU Space = **11 cores** (22 vCPU) on the same socket.


AWS Ground Station Antenna Region
Customer Destination Region
EIP Public IP
Customer Receiver
Dedicated Agent CPU Space
AWS GS Agent
Transmit (Tx) DigIF Dataflow 1
UDP Ingress Port Range 1
Egress Address 1 UDP Socket
DigIF Data
AwsGroundStationAgentEndpoint Listener (e.g. SDR) Dataflow 1
RF Signal
AWS KMS Key Encrypted Forward-Error-Corrected (FEC) Dataflows
Egress Address 2 UDP Socket
DigIF Data
AwsGroundStationAgentEndpoint Listener (e.g. SDR) Dataflow 2
Transmit (Tx) DigIF Dataflow 2
UDP Ingress Port Range 2
/opt/aws/groundstation/etc/aws-gs-agent-config.json

Multiple Dataflows, Multiple Receivers

Example Scenario:

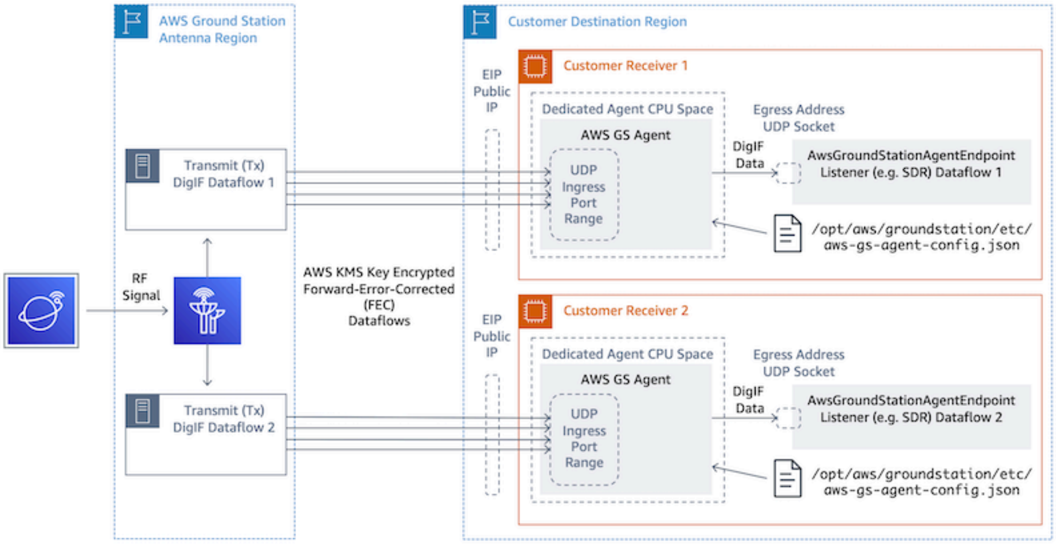
The customer would like to receive two antenna downlinks as DigIF dataflows at different EC2 receiver instances. Both downlinks will be 400MHz.

AwsGroundStationAgentEndpoints:

There will be two `AwsGroundStationAgentEndpoint` resources, one for each dataflow. The endpoints will have a different public IP address (`ingressAddress.socketAddress.name`). There is no restriction on the port values for either `ingressAddress` or `egressAddress` as the dataflows are received on separate infrastructure and will not conflict with each other.

CPU Planning:

- Receiver Instance 1
  - 1 core (2 vCPU) for running the single AWS Ground Station Agent on the instance.
  - 9 cores (18 vCPU) to receive DigIF Dataflow 1 (400MHz lookup in CPU Core Planning table).
  - Total Dedicated Agent CPU Space = **10 cores** (20 vCPU) on the same socket.
- Receiver Instance 2
  - 1 core (2 vCPU) for running the single AWS Ground Station Agent on the instance.
  - 9 cores (18 vCPU) to receive DigIF Dataflow 2 (400MHz lookup in CPU Core Planning table).
  - Total Dedicated Agent CPU Space = **10 cores** (20 vCPU) on the same socket.


AWS Ground Station Antenna Region
Transmit (Tx) DigIF Dataflow 1
RF Signal
Transmit (Tx) DigIF Dataflow 2
AWS KMS Key Encrypted Forward-Error-Corrected (FEC) Dataflows
Customer Destination Region
EIP Public IP
Customer Receiver 1
Dedicated Agent CPU Space
AWS GS Agent
UDP Ingress Port Range
Egress Address UDP Socket
DigIF Data
AwsGroundStationAgentEndpoint Listener (e.g. SDR) Dataflow 1
/opt/aws/groundstation/etc/aws-gs-agent-config.json
EIP Public IP
Customer Receiver 2
Dedicated Agent CPU Space
AWS GS Agent
UDP Ingress Port Range
Egress Address UDP Socket
DigIF Data
AwsGroundStationAgentEndpoint Listener (e.g. SDR) Dataflow 2
/opt/aws/groundstation/etc/aws-gs-agent-config.json

EC2 Instance Selection and CPU Planning

Supported EC2 Instance Types

The AWS Ground Station Agent requires dedicated CPU cores to operate due to the compute intensive data delivery workflows. We support the following instance types. See CPU Core Planning to decide which instance type best suits your use case.

Instance type	Default vCPUs	Default CPU cores
c5.12xlarge	48	24
c5.18xlarge	72	36
c5.24xlarge	96	48
c5n.18xlarge	72	36
c5n.metal	72	36
c6i.32xlarge	128	64
g4dn.12xlarge	48	24
g4dn.16xlarge	64	32
g4dn.metal	96	48
m4.16xlarge	64	32
m5.12xlarge	48	24
m5.24xlarge	96	48
m6i.32xlarge	128	64
p3dn.24xlarge	96	48
p4d.24xlarge	96	48
r5.24xlarge	96	48

Instance type	Default vCPUs	Default CPU cores
r5.metal	96	48
r5n.24xlarge	96	48
r5n.metal	96	48
r6i.32xlarge	128	64

CPU Core Planning

The AWS Ground Station Agent requires dedicated processor cores that share L3 cache for each dataflow. The agent is designed to leverage Hyper-threaded (HT) CPU pairs and requires HT pairs to be reserved for its use. A hyper-threaded pair is a pair of virtual CPUs (vCPU) that are contained within a single core. The following table provides a mapping of dataflow data rate to the required number of cores reserved for the agent for a single dataflow. This table assumes Cascade Lake or newer CPUs and is valid for any supported instance type. If your bandwidth is between entries in the table, select the next highest.

The agent needs an additional reserved core for management and coordination, so the total cores required will be the sum of the cores needed (from the below chart) for each dataflow plus **a single additional core (2 vCPUs)**.

AntennaDownlink Bandwidth (MHz)	Expected VITA-49.2 DigIF Data Rate (Mb/s)	Number of Cores (HT CPU Pairs)	Total vCPU
50	1000	3	6
100	2000	4	8
150	3000	5	10
200	4000	6	12
250	5000	6	12
300	6000	7	14

AntennaDownlink Bandwidth (MHz)	Expected VITA-49.2 DigIF Data Rate (Mb/s)	Number of Cores (HT CPU Pairs)	Total vCPU
350	7000	8	16
400	8000	9	18

Gathering Architecture Information

`lscpu` provides information about the architecture of your system. The basic output shows which vCPUs (labeled as "CPU") belong to which NUMA nodes (and each NUMA node shares an L3 cache). Below we examine a `c5.24xlarge` instance in order to gather the necessary information to configure the AWS Ground Station Agent. This includes useful information like number of vCPUs, cores, and vCPU-to-node association.

```
> lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 96
On-line CPU(s) list: 0-95
Thread(s) per core: 2             <------
Core(s) per socket: 24
Socket(s): 2
NUMA node(s): 2
Vendor ID: GenuineIntel
CPU family: 6
Model: 85
Model name: Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz
Stepping: 7
CPU MHz: 3601.704
BogoMIPS: 6000.01
Hypervisor vendor: KVM
Virtualization type: full
L1d cache: 32K
L1i cache: 32K
L2 cache: 1024K
L3 cache: 36608K
```

```
NUMA node0 CPU(s): 0-23,48-71     <------
NUMA node1 CPU(s): 24-47,72-95    <------
```

Cores dedicated to the AWS Ground Station Agent should include both vCPUs for each assigned core. All cores for a dataflow must exist on the same NUMA node. The `-p` option for the `lscpu` command provides us with the core to CPU associations needed to configure the agent. The relevant fields are CPU (which is what we refer to as the vCPU), Core, and L3 (which indicates which L3 cache is shared by that core). Note that on most Intel processors the NUMA Node is equal to the L3 cache.

Consider the following subset of the `lscpu -p` output for a `c5.24xlarge` (abbreviated and formatted for clarity).

```
CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0   0    0      0     0   0   0  0
1   1    0      0     1   1   1  0
2   2    0      0     2   2   2  0
3   3    0      0     3   3   3  0
...
16  0    0      0     0   0   0  0
17  1    0      0     1   1   1  0
18  2    0      0     2   2   2  0
19  3    0      0     3   3   3  0
```

From the output we can see that Core 0 includes vCPUs 0 and 16, Core 1 includes vCPUs 1 and 17, Core 2 includes vCPUs 2 and 18. In other words the hyper-threaded pairs are: 0 and 16, 1 and 17, 2 and 18.

CPU Assignment Example

As an example, we will use a `c5.24xlarge` instance for a Dual Polarity Wideband downlink at 350MHz. From the table in CPU Core Planning we know that a 350 MHz downlink requires 8 cores (16 vCPUs) for the single data flow. This means that this dual polarity setup using two dataflows requires a total of 16 cores (32 vCPUs) plus one core (2 vCPUs) for the Agent.

We know the `lscpu` output for `c5.24xlarge` includes `NUMA node0 CPU(s): 0-23,48-71` and `NUMA node1 CPU(s): 24-47,72-95`. Since NUMA node0 has more than we need, we will only assign from cores: 0-23 and 48-71.

First, we will select 8 cores for each dataflow that share an L3 cache or NUMA Node. Then we will look up the corresponding vCPUs (labeled “CPU”) in the `lscpu -p` output in [Appendix: `lscpu -p` output (full) for c5.24xlarge](). An example core selection process might look like the following:

- Reserve cores 0-1 for the OS.
- Flow 1: select cores 2-9 which map to vCPUs 2-9 and 50-57.
- Flow 2: select cores 10-17 which map to vCPUs 10-17 and 58-65.
- Agent core: select core 18 which maps to vCPUs 18 and 66.

This results in vCPUs 2-18 and 51-66 so the list to provide the agent is `[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]`. You should ensure your own processes are not running on these CPUs as described in [Running Services and Processes Alongside the AWS Ground Station Agent]().

Note that the specific cores selected in this example are somewhat arbitrary. Other sets of cores would work as long as they satisfy the requirement of all sharing an L3 cache for each dataflow.

Appendix: `lscpu -p` output (full) for c5.24xlarge

```
> lscpu -p
# The following is the parsable format, which can be fed to other
# programs. Each different item in every column has an unique ID
# starting from zero.
# CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0,0,0,0,,0,0,0,0
1,1,0,0,,1,1,1,0
2,2,0,0,,2,2,2,0
3,3,0,0,,3,3,3,0
4,4,0,0,,4,4,4,0
5,5,0,0,,5,5,5,0
6,6,0,0,,6,6,6,0
7,7,0,0,,7,7,7,0
8,8,0,0,,8,8,8,0
9,9,0,0,,9,9,9,0
10,10,0,0,,10,10,10,0
11,11,0,0,,11,11,11,0
12,12,0,0,,12,12,12,0
13,13,0,0,,13,13,13,0
```

```
14,14,0,0,,14,14,14,0
15,15,0,0,,15,15,15,0
16,16,0,0,,16,16,16,0
17,17,0,0,,17,17,17,0
18,18,0,0,,18,18,18,0
19,19,0,0,,19,19,19,0
20,20,0,0,,20,20,20,0
21,21,0,0,,21,21,21,0
22,22,0,0,,22,22,22,0
23,23,0,0,,23,23,23,0
24,24,1,1,,24,24,24,1
25,25,1,1,,25,25,25,1
26,26,1,1,,26,26,26,1
27,27,1,1,,27,27,27,1
28,28,1,1,,28,28,28,1
29,29,1,1,,29,29,29,1
30,30,1,1,,30,30,30,1
31,31,1,1,,31,31,31,1
32,32,1,1,,32,32,32,1
33,33,1,1,,33,33,33,1
34,34,1,1,,34,34,34,1
35,35,1,1,,35,35,35,1
36,36,1,1,,36,36,36,1
37,37,1,1,,37,37,37,1
38,38,1,1,,38,38,38,1
39,39,1,1,,39,39,39,1
40,40,1,1,,40,40,40,1
41,41,1,1,,41,41,41,1
42,42,1,1,,42,42,42,1
43,43,1,1,,43,43,43,1
44,44,1,1,,44,44,44,1
45,45,1,1,,45,45,45,1
46,46,1,1,,46,46,46,1
47,47,1,1,,47,47,47,1
48,0,0,0,,0,0,0,0
49,1,0,0,,1,1,1,0
50,2,0,0,,2,2,2,0
51,3,0,0,,3,3,3,0
52,4,0,0,,4,4,4,0
53,5,0,0,,5,5,5,0
54,6,0,0,,6,6,6,0
55,7,0,0,,7,7,7,0
56,8,0,0,,8,8,8,0
57,9,0,0,,9,9,9,0
```

```
58,10,0,0,,10,10,10,0
59,11,0,0,,11,11,11,0
60,12,0,0,,12,12,12,0
61,13,0,0,,13,13,13,0
62,14,0,0,,14,14,14,0
63,15,0,0,,15,15,15,0
64,16,0,0,,16,16,16,0
65,17,0,0,,17,17,17,0
66,18,0,0,,18,18,18,0
67,19,0,0,,19,19,19,0
68,20,0,0,,20,20,20,0
69,21,0,0,,21,21,21,0
70,22,0,0,,22,22,22,0
71,23,0,0,,23,23,23,0
72,24,1,1,,24,24,24,1
73,25,1,1,,25,25,25,1
74,26,1,1,,26,26,26,1
75,27,1,1,,27,27,27,1
76,28,1,1,,28,28,28,1
77,29,1,1,,29,29,29,1
78,30,1,1,,30,30,30,1
79,31,1,1,,31,31,31,1
80,32,1,1,,32,32,32,1
81,33,1,1,,33,33,33,1
82,34,1,1,,34,34,34,1
83,35,1,1,,35,35,35,1
84,36,1,1,,36,36,36,1
85,37,1,1,,37,37,37,1
86,38,1,1,,38,38,38,1
87,39,1,1,,39,39,39,1
88,40,1,1,,40,40,40,1
89,41,1,1,,41,41,41,1
90,42,1,1,,42,42,42,1
91,43,1,1,,43,43,43,1
92,44,1,1,,44,44,44,1
93,45,1,1,,45,45,45,1
94,46,1,1,,46,46,46,1
95,47,1,1,,47,47,47,1
```

Installing the agent

The AWS Ground Station Agent can be installed in the following ways:

1. AWS CloudFormation template (recommended).
2. Manual install on Amazon EC2.

Using CloudFormation template

The EC2 data delivery CloudFormation template creates the required AWS resources to deliver data to your EC2 instance. This AWS CloudFormation template uses the AWS Ground Station managed AMI that has the AWS Ground Station Agent pre-installed. The created EC2 instance's boot script then populates the agent configuration file and applies the necessary performance tuning (EC2 Instance Performance Tuning).

Step 1: Create AWS Resources

Create your AWS resources stack using template Direct Broadcast Satellite Wideband DigIF Template (Wideband).

Step 2: Check Agent Status

By default the the agent is configured and active (started). In order to check the agent status you can connect to the EC2 instance (SSH or SSM Session Manager) and see AWS Ground Station Agent Status.

Manual install on EC2

While Ground Station recommends using CloudFormation templates to provision your AWS Resources there may be use cases where the standard template may not suffice. For such cases we recommend that you customize the template to suit your needs. If that still does not meet your requirements, you can manually create your AWS resources and install the agent.

Step 1: Create AWS Resources

See Manually Creating and Configuring Resources for instructions to set up the AWS resources required for a contact manually.

The **AwsGroundStationAgentEndpoint** resource defines an endpoint for receiving a DigIF dataflow via AWS Ground Station Agent and is a critical to taking a successful contact. While the API documentation is located in the API Reference, this section will briefly discuss concepts relevant to the AWS Ground Station Agent.

The endpoint's `ingressAddress` is where the AWS Ground Station Agent will receive AWS KMS encrypted UDP traffic from the Antenna. The `socketAddress` `name` is the public IP of the EC2 instance (from the attached EIP). The `portRange` should be at least 300 contiguous ports in a range that has been reserved from any other usage. See Reserve Ingress Ports - Impacts Network for instructions. These ports must be configured to allow UDP ingress traffic on the security group for the VPC where the receiver instance is running.

The endpoint's `egressAddress` is where the Agent will hand off the DigIF dataflow to the customer. The customer should have an application (e.g. SDR) receiving the data over a UDP socket at this location.

Step 2: Create EC2 instance

The following AMIs are supported:

1. AWS Ground Station AMI - `groundstation-al2-gs-agent-ami-*` where * is the date the AMI was built - comes with agent installed (recommended).
2. `amzn2-ami-kernel-5.10-hvm-x86_64-gp2`.

Step 3: Download and install agent

> **Note**
>
> Steps in this section must be completed if you did **not** choose the AWS Ground Station Agent AMI in the previous step.

Download agent

The AWS Ground Station Agent is available from region specific S3 buckets and can be downloaded onto support EC2 instances using the AWS command line (CLI) from `s3://groundstation-wb-digif-software-${AWS::Region}/aws-groundstation-agent/latest/amazon_linux_2_x86_64/aws-groundstation-agent.rpm` where ${AWS::Region} refers to one of the supported AWS Ground Station Console and Data Delivery Regions.

Example: Download the latest rpm version from AWS region us-east-2 locally to the /tmp folder.

```
aws s3 --region us-east-2 cp s3://groundstation-wb-digif-software-us-east-2/aws-
groundstation-agent/latest/amazon_linux_2_x86_64/aws-groundstation-agent.rpm /tmp
```

If you need to download a specific version of the AWS Ground Station Agent, you can download it from the version specific folder in the S3 bucket.

Example: Download version 1.0.2716.0 of the rpm from AWS region us-east-2 locally to the /tmp folder.

```
aws s3 --region us-east-2 cp s3://groundstation-wb-digif-software-us-east-2/aws-
groundstation-agent/1.0.2716.0/amazon_linux_2_x86_64/aws-groundstation-agent.rpm /tmp
```

Note

If you want to confirm the RPM you downloaded was vended by AWS Ground Station, follow the instructions for RPM Installation Validation.

Install agent

```
sudo yum install ${MY_RPM_FILE_PATH}

Example: Assumes agent is in the "/tmp" directory
sudo yum install /tmp/aws-groundstation-agent.rpm
```

Step 4: Configure the agent

After installing the agent you must update the agent configuration file. See Configuring the agent.

Step 5: Apply Performance Tuning

AWS Ground Station Agent AMI: If you chose the AWS Ground Station Agent AMI in the previous step then apply the following performance tunings.

- Tune Hardware Interrupts and Receive Queues - Impacts CPU and Network

- Reserve Ingress Ports - Impacts Network
- Reboot

Other AMIs: If you chose any other AMI in the previous step then apply all tunings listed under EC2 Instance Performance Tuning and Reboot the instance.

Step 6: Manage the agent

In order to start, stop and check agent status see Managing the agent.

Managing the agent

AWS Ground Station Agent provides the following capabilities for configuring, starting, stopping, upgrading, downgrading and uninstalling the agent using built-in Linux command tooling.

Topics

- AWS Ground Station Agent Configuration
- AWS Ground Station Agent Start
- AWS Ground Station Agent Stop
- AWS Ground Station Agent Upgrade
- AWS Ground Station Agent Downgrade
- AWS Ground Station Agent Uninstall
- AWS Ground Station Agent Status
- AWS Ground Station Agent RPM Info

AWS Ground Station Agent Configuration

Navigate to `/opt/aws/groundstation/etc`, which should contain a single file named aws-gs-agent-config.json. See Agent Config File

AWS Ground Station Agent Start

```
#start
sudo systemctl start aws-groundstation-agent

#check status
systemctl status aws-groundstation-agent
```

Should produce output showing agent is **active**.

```
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: active (running) since Tue 2023-03-14 00:39:08 UTC; 1 day 13h ago
Docs: https://aws.amazon.com/ground-station/
Main PID: 8811 (aws-gs-agent)
CGroup: /system.slice/aws-groundstation-agent.service
##8811 /opt/aws/groundstation/bin/aws-gs-agent production
```

AWS Ground Station Agent Stop

```
#stop
sudo systemctl stop aws-groundstation-agent

#check status
systemctl status aws-groundstation-agent
```

Should produce output showing agent is **inactive** (stopped).

```
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: inactive (dead) since Thu 2023-03-09 15:35:08 UTC; 6min ago
Docs: https://aws.amazon.com/ground-station/
Process: 84182 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=0/SUCCESS)
Main PID: 84182 (code=exited, status=0/SUCCESS)
```

AWS Ground Station Agent Upgrade

1. Download the latest version of the agent. See Download agent.
2. Stop the agent.

   ```
   #stop
   sudo systemctl stop aws-groundstation-agent

   #confirm inactive (stopped) state
   systemctl status aws-groundstation-agent
   ```

3. Update the agent.

   ```
   sudo yum update ${MY_RPM_FILE_PATH}

   # check the new version has been installed correctly by comparing the agent version
    with the starting agent version
   yum info aws-groundstation-agent

   # reload the systemd configuration
   sudo systemctl daemon-reload

   # restart the agent
   sudo systemctl restart aws-groundstation-agent

   # check agent status
   systemctl status aws-groundstation-agent
   ```

AWS Ground Station Agent Downgrade

1. Download the agent version you need. See Download agent.
2. Downgrade the agent.

```
# get the starting agent version
yum info aws-groundstation-agent

# stop the agent service
sudo systemctl stop aws-groundstation-agent

# downgrade the rpm
sudo yum downgrade ${MY_RPM_FILE_PATH}

# check the new version has been installed correctly by comparing the agent version
 with the starting agent version
yum info aws-groundstation-agent

# reload the systemd configuration
sudo systemctl daemon-reload

# restart the agent
sudo systemctl restart aws-groundstation-agent

# check agent status
systemctl status aws-groundstation-agent
```

AWS Ground Station Agent Uninstall

Uninstalling the agent will rename /opt/aws/groundstation/etc/aws-gs-agent-config.json to /opt/aws/groundstation/etc/aws-gs-agent-config.json.rpmsave. Installing the agent again on the same instance again will write default values for aws-gs-agent-config.json and will need to be updated with the correct values corresponding to your AWS resources. See Agent Config File.

```
sudo yum remove aws-groundstation-agent
```

AWS Ground Station Agent Status

The agent status is either **active** (agent is running) or **inactive** (agent is stopped).

```
systemctl status aws-groundstation-agent
```

An example output shows that the agent is installed, **inactive** state (stopped) and **enabled** (start service on boot).

```
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: inactive (dead) since Thu 2023-03-09 15:35:08 UTC; 6min ago
Docs: https://aws.amazon.com/ground-station/
Process: 84182 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=0/SUCCESS)
Main PID: 84182 (code=exited, status=0/SUCCESS)
```

AWS Ground Station Agent RPM Info

```
yum info aws-groundstation-agent
```

The output is as follows:

Note

"Version" might be different based on latest agent published version.

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Installed Packages
Name         : aws-groundstation-agent
Arch         : x86_64
Version      : 1.0.2677.0
Release      : 1
Size         : 51 M
Repo         : installed
Summary      : Client software for AWS Ground Station
URL          : https://aws.amazon.com/ground-station/
```

```
License     : Proprietary
Description : This package provides client applications for use with AWS Ground Station
```

Configuring the agent

After installing the agent you must update the agent configuration file at `/opt/aws/groundstation/etc/aws-gs-agent-config.json`.

Agent Config File

Example

```
{
  "capabilities": [
    "arn:aws:groundstation:eu-central-1:123456789012:dataflow-endpoint-group/
bb6c19ea-1517-47d3-99fa-3760f078f100"
  ],
  "device": {
  "privateIps": [
    "127.0.0.1"
  ],
  "publicIps": [
    "1.2.3.4"
  ],
  "agentCpuCores":
 [ 24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,72,73,74,75,76,77,78,79,80,81
}
```

Field Breakdown

capabilities

Capabilities are specified as Dataflow Endpoint Group Amazon Resource Names.

Required: True

Format: String Array

- Values: capability ARNs → String

Examples:

```
"capabilities": [
    "arn:aws:groundstation:${AWS::Region}:${AWS::AccountId}:dataflow-endpoint-group/
${DataflowEndpointGroupId}"
]
```

device

This field contains additional fields necessary to enumerate the current EC2 “device”.

Required: True

Format: Object

Members:

- privateIps
- publicIps
- agentCpuCores
- networkAdapters

privateIps

This field is currently not used, but is included for future use cases. If no value is included, it will default to [“127.0.0.1”]

Required: False

Format: String Array

- Values: IP Addresses → String

Example:

```
"privateIps": [
    "127.0.0.1"
],
```

publicIps

Elastic IP (EIP) per dataflow endpoint group.

Required: True

Format: String Array

- Values: IP Addresses → String

Example:

```
"publicIps": [
    "9.8.7.6"
],
```

agentCPUCores

This specifies which virtual cores are reserved for the aws-gs-agent process. See CPU Core Planning for requirements to appropriately set this value.

Required: True

Format: Int Array

- Values: Core Numbers → int

Example:

```
"agentCpuCores": [
    
 24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,72,73,74,75,76,77,78,79,80,81,8
]
```

networkAdapters

This corresponds to the ethernet adapters, or interfaces attached to ENIs, that will receive data.

Required: False

Format: String Array

- Values: ethernet adapter names (can find them by running `ifconfig`)

Example:

```
"networkAdapters": [
    "eth0"
]
```

EC2 Instance Performance Tuning

> **Note**
>
> If you provisioned your AWS resources using CloudFormation templates these tunings are automatically applied. If you used an AMI or manually created your EC2 instance then these performance tunings must be applied to achieve the most reliable performance.

Remember to reboot your instance after applying any tuning(s).

Topics

- Reboot

Tune Hardware Interrupts and Receive Queues - Impacts CPU and Network

This section configures CPU core usage of systemd, SMP IRQs, Receive Packet Steering (RPS) and Receive Flow Steering (RFS). See Appendix: Recommended Parameters for Interrupt/RPS Tune for a set of recommended settings based on the instance type you are using.

1. Pin systemd processes away from agent CPU cores.
2. Route hardware interrupt requests away from agent CPU cores.
3. Configure RPS to prevent the hardware queue of a single network interface card from becoming a bottleneck in network traffic.
4. Configure RFS to increase the CPU cache hit rate and thereby reduce network latency.

The `set_irq_affinity.sh` script provided by the RPM configures all the above for you. Add to crontab so it is applied on each boot:

```
echo "@reboot sudo /opt/aws/groundstation/bin/set_irq_affinity.sh
 '${interrupt_core_list}' '${rps_core_mask}' >> /var/log/user-data.log 2>&1" >>/var/
spool/cron/root
```

- Replace `interrupt_core_list` with cores reserved for the kernel and OS - typically the first and second along with hyper-threaded core pairs. This should not overlap with the cores selected above. (Ex: ‘0,1,48,49’ for a hyper-threaded, 96-CPU instance).
- `rps_core_mask` is a hexadecimal bit mask specifying which CPUs should be processing incoming packets, with each digit representing 4 CPUs. It must also be comma separated every 8 characters starting from the right. It is recommended to allow all CPUs and let caching handle the balancing.
    - To see the list of recommended parameters for each instance type, refer to Appendix: Recommended Parameters for Interrupt/RPS Tune.
- Example for 96-CPU instance:

```
echo "@reboot sudo /opt/aws/groundstation/bin/set_irq_affinity.sh '0,1,48,49'
 'ffffffff,ffffffff,ffffffff' >> /var/log/user-data.log 2>&1" >>/var/spool/cron/root
```

Tune Rx Interrupt Coalescing - Impacts Network

Interrupt coalescing helps prevent flooding the host system with too many interrupts and helps increase network throughput. With this configuration, packets are collected and one single interrupt is generated every 128 microseconds. Add to crontab so it is applied on each boot:

```
echo "@reboot sudo ethtool -C ${interface} rx-usecs 128 tx-usecs 128 >>/var/log/user-
data.log 2>&1" >>/var/spool/cron/root
```

- Replace `interface` with the network interface (ethernet adapter) configured to receive data. Typically this is `eth0` as thats the default network interface assigned for an EC2 instance.

Tune Rx Ring Buffer - Impacts Network

Increase the number of ring entries for the Rx ring buffer to prevent packet drops or overruns during bursty connections. Add to the crontab so it is correctly set on each boot:

```
echo "@reboot sudo ethtool -G ${interface} rx 16384 >>/var/log/user-data.log 2>&1" >>/
var/spool/cron/root
```

- Replace `interface` with the network interface (ethernet adapter) configured to receive data. Typically this is `eth0` as thats the default network interface assigned for an EC2 instance.
- If setting up a c6i.32xlarge instance, command needs to be modified to set the ring buffer to `8192`, instead of `16384`.

Tune CPU C-State - Impacts CPU

Set the CPU C-state to prevent idling which can cause lost packets during the start of a contact. Requires instance reboot.

```
echo "GRUB_CMDLINE_LINUX_DEFAULT=\"console=tty0 console=ttyS0,115200n8
 net.ifnames=0 biosdevname=0 nvme_core.io_timeout=4294967295 intel_idle.max_cstate=1
 processor.max_cstate=1 max_cstate=1\"" >/etc/default/grub
echo "GRUB_TIMEOUT=0" >>/etc/default/grub
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Reserve Ingress Ports - Impacts Network

Reserve all ports in your `AwsGroundStationAgentEndpoint`'s ingress address port range to prevent conflicts with kernel usage. Port usage conflict will lead to contact and data delivery failure.

```
echo "net.ipv4.ip_local_reserved_ports=${port_range_min}-${port_range_max}" >> /etc/
sysctl.conf
```

- Example: `echo "net.ipv4.ip_local_reserved_ports=42000-43500" >> /etc/sysctl.conf`.

Reboot

After all tunings are applied successfully, reboot the instance for the tunings to take effect.

```
sudo reboot
```

Appendix: Recommended Parameters for Interrupt/RPS Tune

This section determines the recommended parameter values for use in tuning section Tune Hardware Interrupts and Receive Queues - Impacts CPU and Network.

Family	Instance Type	${interrupt_core_list}	${rps_core_mask}
c6i	- c6i.32xlarge	- 0,1,64,65	- ffffffff,ffffffff,ffffffff,ffffffff
c5	- c5.24xlarge - c5.18xlarge - c5.12xlarge	- 0,1,48,49 - 0,1,36,37 - 0,1,24,25	- ffffffff,ffffffff,ffffffff - ff,ffffffff,ffffffff - ffff,ffffffff
c5n	- c5n.metal - c5n.18xlarge	- 0,1,36,37 - 0,1,36,37	- ff,ffffffff,ffffffff - ff,ffffffff,ffffffff
m5	- m5.24xlarge - m5.12xlarge	- 0,1,48,49 - 0,1,24,25	- ffffffff,ffffffff,ffffffff - ffff,ffffffff
r5	- r5.metal - r5.24xlarge	- 0,1,48,49 - 0,1,48,49	- ffffffff,ffffffff,ffffffff - ffffffff,ffffffff,ffffffff
r5n	- r5n.metal - r5n.24xlarge	- 0,1,48,49 - 0,1,48,49	- ffffffff,ffffffff,ffffffff

Family	Instance Type	${interrupt_core_list}	${rps_core_mask}
			• ffffffff, ffffffff, ffffffff
g4dn	• g4dn.metal • g4dn.16xlarge • g4dn.12xlarge	• 0,1,48,49 • 0,1,32,33 • 0,1,24,25	• ffffffff, ffffffff, ffffffff • ffffffff, ffffffff • ffff,ffffffff
p4d	• p4d.24xlarge	• 0,1,48,49	• ffffffff, ffffffff, ffffffff
p3dn	• p3dn.24xlarge	• 0,1,48,49	• ffffffff, ffffffff, ffffffff

Prepare to take a DigIF contact

1. Review CPU Core Planning for desired dataflows, and provide a list of cores the agent can use. See CPU Core Planning.
2. Review the AWS Ground Station Agent configuration file. See AWS Ground Station Agent Configuration.
3. Confirm that the necessary performance tuning is applied. See EC2 Instance Performance Tuning.
4. Confirm you are following all the called out best practices. See Best practices.
5. Confirm that the AWS Ground Station Agent is started prior to the scheduled contact start time via:

```
systemctl status aws-groundstation-agent
```

6. Confirm that the AWS Ground Station Agent is healthy prior to the scheduled contact start time via:

```
aws groundstation get-dataflow-endpoint-group --dataflow-endpoint-group-id
 ${DATAFLOW-ENDPOINT-GROUP-ID} --region ${REGION}
```

   Verify that the `agentStatus` of your `awsGroundStationAgentEndpoint` is ACTIVE and the `auditResults` is HEALTHY.

Best practices

Best EC2 practices

Follow current EC2 best practices and ensure sufficient data storage availability.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-best-practices.html

Linux Scheduler

The Linux scheduler can re-order packets on UDP sockets if the corresponding processes are not pinned to a specific core. Any thread sending or receiving UDP data should pin itself to a specific core for the duration of data transmission.

AWS Ground Station Managed Prefix List

It is recommended to utilize the `com.amazonaws.global.groundstation` AWS-managed prefix list when specifying the network rules to allow communication from the Antenna. See Working with AWS Managed Prefix Lists for more information about AWS Managed Prefix Lists.

Single contact limitation

The AWS Ground Station Agent supports multiple streams per contact, but only supports a single contact at a time. To prevent scheduling issues, do not share an instance across multiple dataflow endpoint groups. If a single agent configuration is associated with multiple different DFEG ARNs, it will fail to register.

Running Services and Processes Alongside the AWS Ground Station Agent

When launching services and processes on the same EC2 Instance as the AWS Ground Station Agent, it is important to bind them to vCPUs not in use by the AWS Ground Station Agent and Linux kernel as this can cause bottlenecks and even data loss during contacts. This concept of binding to specific vCPUs is known as affinity.

Cores to avoid:

- `agentCpuCores` from Agent Config File
- `interrupt_core_list` from Tune Hardware Interrupts and Receive Queues - Impacts CPU and Network.
  - Default values can be found from Appendix: Recommended Parameters for Interrupt/RPS Tune

As an example using a `c5.24xlarge` instance

If you specified

```
"agentCpuCores": [24,25,26,27,72,73,74,75]"
```

and ran

```
echo "@reboot sudo /opt/aws/groundstation/bin/set_irq_affinity.sh
'0,1,48,49' 'ffffffff,ffffffff,ffffffff' >> /var/log/user-data.log 2>&1"
>>/var/spool/cron/root
```

then avoid the following cores:

```
0,1,24,25,26,27,48,49,72,73,74,75
```

Affinitizing Services (systemd)

Newly launched services will automatically affinitize to the `interrupt_core_list` mentioned previously. If your launched services' use-case requires additional cores, or needs less congested cores, follow this section.

Check what affinity your service is currently configured to with command:

```
systemctl show --property CPUAffinity <service name>
```

If you see an empty value like `CPUAffinity=`, that means it will likely use the default cores from the above command `...bin/set_irq_affinity.sh <using the cores here> ...`

To override and set a specific affinity find the location of the service file by running:

```
systemctl show -p FragmentPath <service name>
```

Open and modify the file (using `vi`, `nano`, etc.) and put the `CPUAffinity=<core list>` in the `[Service]` section like:

```
[Unit]
...

[Service]
...
CPUAffinity=2,3

[Install]
...
```

Save the file and restart the service to apply the affinity with:

```
systemctl daemon-reload
systemctl restart <service name>

# Additionally confirm by re-running
systemctl show --property CPUAffinity <service name>
```

For more information visit: Red Hat Enterprise Linux 8 - Managing, monitoring, and updating the kernel - Chapter 27. Configuring CPU Affinity and NUMA policies using systemd.

Affinitizing Processes (scripts)

It is highly recommended for newly launched scripts and processes to be manually affinitized as the default Linux behavior will allow them to use any core on the machine.

To avoid core conflicts for any running processes (such as python, bash scripts, etc.), launch the process with:

```
taskset -c <core list> <command>
# Example: taskset -c 8 ./bashScript.sh
```

If the process is already running, use commands such as `pidof`, `top`, or `ps` to find the Process ID (PID) of the specific process. With the PID you can see current affinity with:

```
taskset -p <pid>
```

and can modify it with:

```
taskset -p <core mask> <pid>
# Example: taskset -p c 32392 (which sets it to cores 0xc -> 0b1100 -> cores 2,3)
```

For more information on taskset see taskset - Linux man page

Troubleshooting

Agent fails to start

The AWS Ground Station Agent may fail to start due to several reasons, but the most common scenario might be a misconfigured agent configuration file. After starting the agent (see AWS Ground Station Agent Start) you might get a status such as:

```
#agent is automatically retrying a restart
aws-groundstation-agent.service - aws-groundstation-agent
```

```
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: activating (auto-restart) (Result: exit-code) since Fri 2023-03-10 01:48:14
 UTC; 23s ago
Docs: https://aws.amazon.com/ground-station/
Process: 43038 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=101)
Main PID: 43038 (code=exited, status=101)

#agent has failed to start
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: failed (Result: start-limit) since Fri 2023-03-10 01:50:15 UTC; 13s ago
Docs: https://aws.amazon.com/ground-station/
Process: 43095 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=101)
Main PID: 43095 (code=exited, status=101)
```

Troubleshooting

```
sudo journalctl -u aws-groundstation-agent | grep -i -B 3 -A 3 'Loading Config' | tail
 -6
```

might result in an output of:

```
launch-aws-gs-agent[43095]: Running with options Production(ProductionOptions
 { endpoint: None, region: None })
launch-aws-gs-agent[43095]: Loading Config
launch-aws-gs-agent[43095]: System has 96 logical cores
systemd[1]: aws-groundstation-agent.service: main process exited, code=exited,
 status=101/n/a
systemd[1]: Unit aws-groundstation-agent.service entered failed state.
```

Failure to start the agent after “Loading Config” indicates an issue with the agent configuration. See Agent Config File to verify your agent configuration.

AWS Ground Station Agent Logs

AWS Ground Station Agent writes information about contact executions, errors, and health status to log files on the instance running the agent. You can view the log files by manually connecting to an instance.

You can view agent logs in the following location.

```
/var/log/aws/groundstation
```

No Contacts Available

Scheduling contacts requires a healthy AWS Ground Station Agent. Please confirm that your AWS Ground Station Agent has started and that it is healthy by querying the AWS Ground Station API via get-dataflow-endpoint-group:

```
aws groundstation get-dataflow-endpoint-group --dataflow-endpoint-group-id ${DATAFLOW-
ENDPOINT-GROUP-ID} --region ${REGION}
```

Verify that the `agentStatus` of your `awsGroundStationAgentEndpoint` is ACTIVE and the `auditResults` is HEALTHY.

Getting Support

Contact the Ground Station team through AWS Support.

1. Provide `contact_id` for any impacted contacts. The AWS Ground Station team cannot investigate a specific contact without this information.
2. Provide details around all troubleshooting steps already taken.
3. Provide any error messages found while running the commands in our troubleshooting guidance.

Agent Release Notes

Latest Agent Version

Version 1.0.3555.0

Release Date: 03/27/2024

End of Support Date: 08/31/2024

RPM Checksums:

- SHA256: `108f3aceb00e5af549839cd766c56149397e448a6e1e1429c89a9eebb6bc0fc1`
- MD5: `65b72fa507fb0af32651adbb18d2e30f`

Changes:

- Add Agent metric for selected executable version during tasking startup.
- Add config file support for avoiding specific executable versions when other versions are available.
- Add network and routing diagnostics.
- Additional security features.
- Fix issue where some metric reporting errors were written to stdout/journal instead of log file.
- Gracefully handle network unreachable socket errors.
- Measure packet loss and latency between source and destination agents.
- Release aws-gs-datapipe version 2.0 to support new protocol features and the ability to transparently upgrade contacts to the new protocol.

Deprecated Agent Versions

Version 1.0.2942.0

Release Date: 06/26/2023

End of Support Date: 05/31/2024

RPM Checksums:

- SHA256: `7d94b642577504308a58bab28f938507f2591d4e1b2c7ea170b77bea97b5a9b6`
- MD5: `661ff2b8f11aba5d657a6586b56e0d8f`

Changes:

- Added error logs for when Agent RPM is updated on disk and needs Agent restart for changes to take effect.
- Added network tuning validation to ensure Agent user guide tuning steps are followed and applied correctly.
- Fix bug that caused erroneous warnings in Agent logs about log archival.
- Improved packet loss detection.
- Updated Agent install to prevent install or upgrade of the RPM if the Agent is already running.

Version 1.0.2716.0

Release Date: 03/15/2023

End of Support Date: 05/31/2024

RPM Checksums:

- SHA256: `cb05b6a77dfcd5c66d81c0072ac550affbcefefc372cc5562ee52fb220844929`
- MD5: `65266490c4013b433ec39ee50008116c`

Changes:

- Enable uploading logs when Agent experiences failures during tasking.
- Fix linux compatability bug in provided network tuning scripts.

Version 1.0.2677.0

Release Date: 02/15/2023

End of Support Date: 05/31/2024

RPM Checksums:

- SHA256: `77cfe94acb00af7ca637264b17c9b21bd7afdc85b99dffdd627aec9e99397489`
- MD5: `b8533be7644bb4d12ab84de21341adac`

Changes:

- First generally available Agent release.

RPM Installation Validation

The latest RPM version, MD5 hash validated from RPM, and SHA256 hash using sha256sum are shown below. These values, combined, can be used to validate the RPM version being used for the ground station agent.

Latest Agent Version

Version 1.0.3555.0

Release Date: 03/27/2024

End of Support Date: 08/31/2024

RPM Checksums:

- SHA256: `108f3aceb00e5af549839cd766c56149397e448a6e1e1429c89a9eebb6bc0fc1`
- MD5: `65b72fa507fb0af32651adbb18d2e30f`

Changes:

- Add Agent metric for selected executable version during tasking startup.
- Add config file support for avoiding specific executable versions when other versions are available.
- Add network and routing diagnostics.
- Additional security features.
- Fix issue where some metric reporting errors were written to stdout/journal instead of log file.
- Gracefully handle network unreachable socket errors.
- Measure packet loss and latency between source and destination agents.

- Release aws-gs-datapipe version 2.0 to support new protocol features and the ability to transparently upgrade contacts to the new protocol.

Verify The RPM

Tools that you will need to be able to verify this RPM installation are:

- sha256sum
- rpm

Both tools come by default on Amazon Linux 2. These tools will help to validate that the RPM you are using is the correct version. First download the latest RPM from the S3 bucket (see Download agent for instructions on downloading the RPM). Once this file is downloaded, there will be a few things to check:

- Calculate the sha256sum of the RPM file. Perform the following action from the command line of the compute instance that you are using:

  ```
  sha256sum aws-groundstation-agent.rpm
  ```

  Take this value and compare it to the table above. This shows that the RPM file that is downloaded is a valid file to use that AWS Ground Station has vended out to customers. If the hashes do not match, do not install the RPM, and delete it from the compute instance.
- Check the MD5 hash of the file as well, to ensure that the RPM has not been compromised. To do this, use the RPM command line tool by running the following command:

  ```
  rpm -Kv ./aws-groundstation-agent.rpm
  ```

  Validate that the MD5 hash listed here is the same as the MD5 hash of the version that is in the table above. Once both of these hashes have been validated against this table that is listed within AWS Docs, the customer can be ensured that the RPM that was downloaded and installed is the safe and uncompromised version of the RPM.

Listing and Reserving Contacts

You can enter satellite data, identify antenna locations, communicate, and schedule antenna time for selected satellites by using the AWS Ground Station console or AWS CLI. You can review, cancel, and reschedule contact reservations up to eight days prior to scheduled time. In addition, you can view the details of your reserved minutes pricing plan if you are using the AWS Ground Station reserved minutes pricing model.

AWS Ground Station supports cross-region data delivery. The dataflow endpoint configs that are part of the mission profile you select determine to which region(s) the data is delivered. For more information about using cross-region data delivery, see Using Cross Region Data Delivery Service.

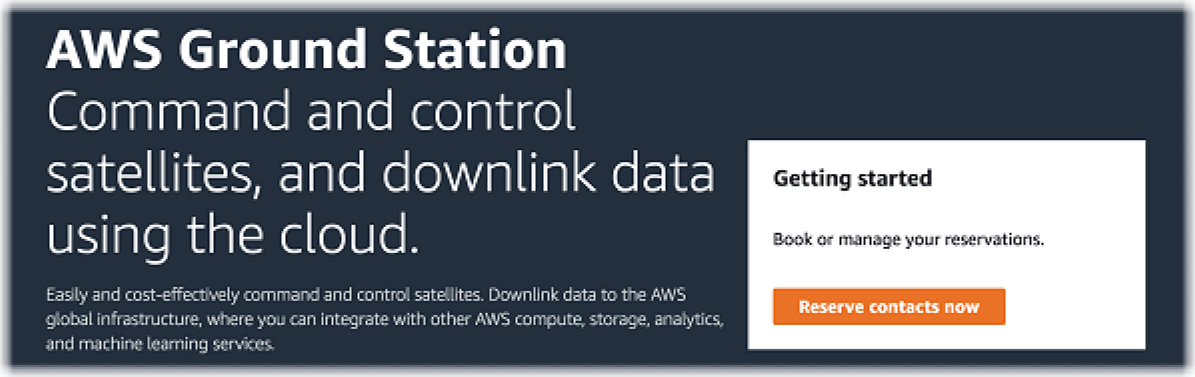
To schedule contacts, your resources must be configured. If you have not configured your resources, see Getting Started.

Topics

- Using the Ground Station Console
- Reserving and Managing Contacts with AWS CLI

Using the Ground Station Console

You can use the AWS Ground Station console to reserve, view, and cancel contact reservations. To use the AWS Ground Station console, open the AWS Ground Station console and choose **Reserve contacts now**.



AWS Ground Station

Command and control satellites, and downlink data using the cloud.

Easily and cost-effectively command and control satellites. Downlink data to the AWS global infrastructure, where you can integrate with other AWS compute, storage, analytics, and machine learning services.

Getting started

Book or manage your reservations.

Use the following topics to use the AWS Ground Station console to reserve, view, and cancel contacts.

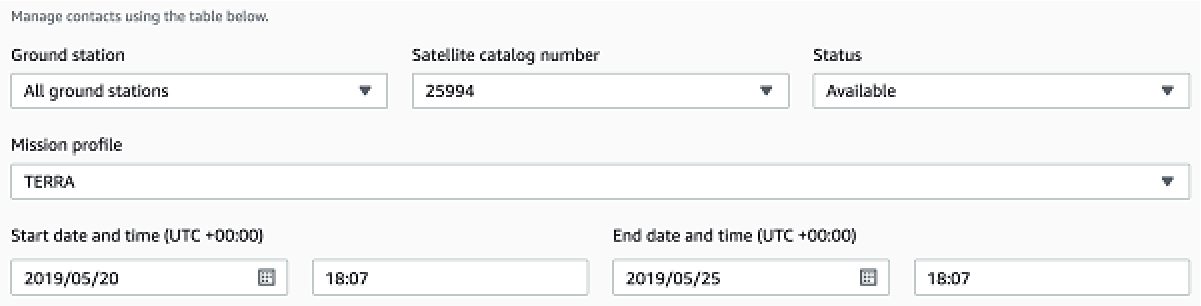
Topics

- Reserve a Contact
- View Scheduled and Completed Contacts
- Cancelling Contacts
- Naming Satellites

Reserve a Contact

After accessing the AWS Ground Station console, use your configured resources to reserve contacts in the **Contact management** table.

1. In the **Contact management** table, choose the parameters you want to use to search for available contacts. Ensure that you are viewing **Available** contacts by using the **Status** filter.



Manage contacts using the table below.

Ground station: All ground stations

Satellite catalog number: 25994

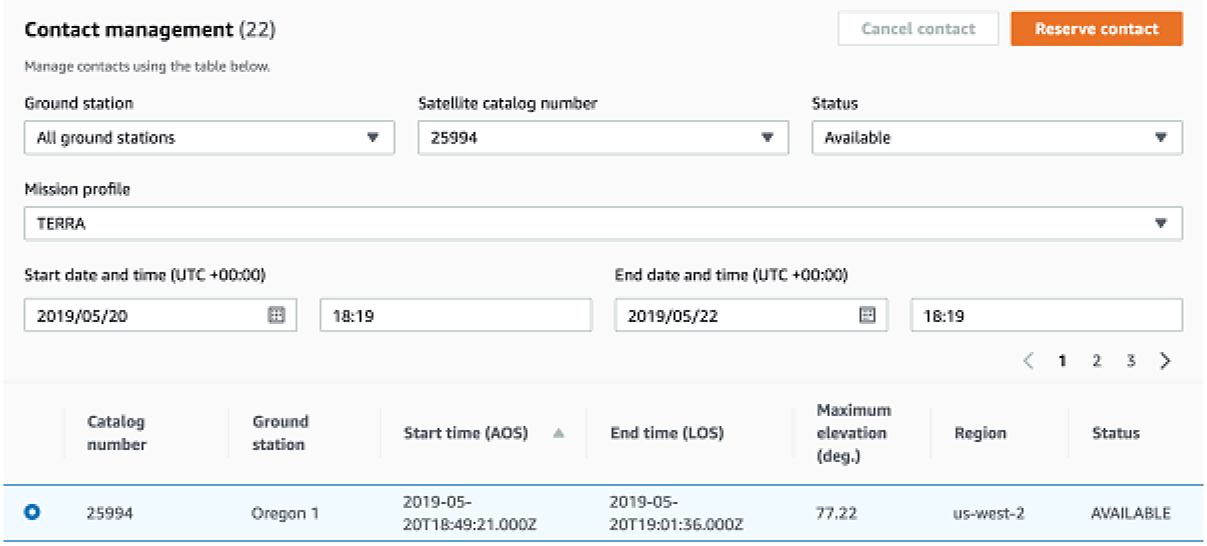
Status: Available

Mission profile: TERRA

Start date and time (UTC +00:00): 2019/05/20 18:07

End date and time (UTC +00:00): 2019/05/25 18:07

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



Contact management (22)

Manage contacts using the table below.

Ground station: All ground stations

Satellite catalog number: 25994

Status: Available

Mission profile: TERRA

Start date and time (UTC +00:00): 2019/05/20 18:19

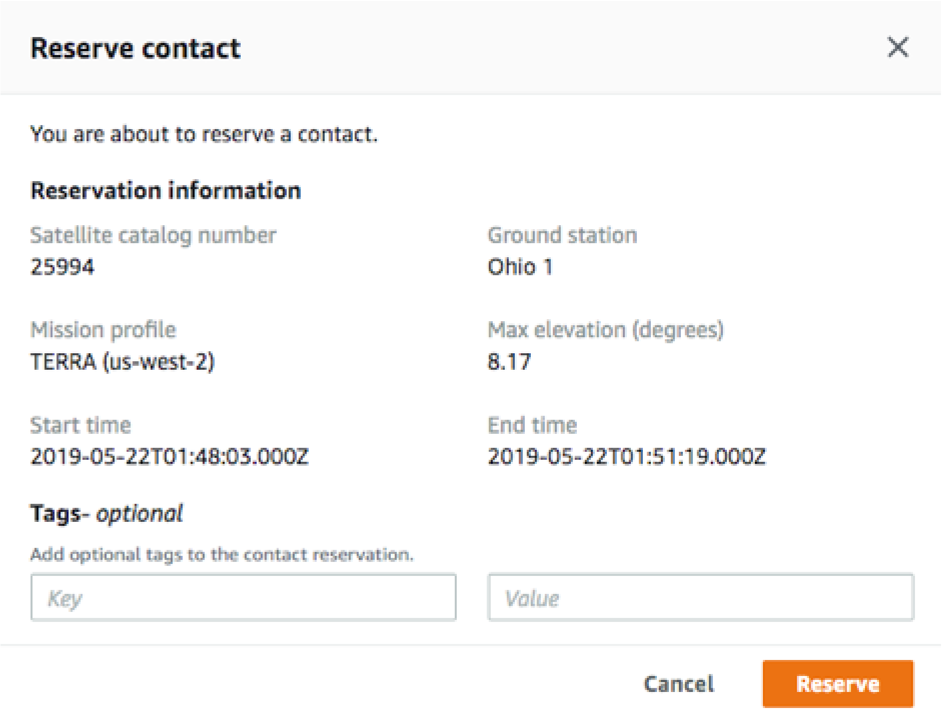
End date and time (UTC +00:00): 2019/05/22 18:19

	Catalog number	Ground station	Start time (AOS) ▲	End time (LOS)	Maximum elevation (deg.)	Region	Status
●	25994	Oregon 1	2019-05-20T18:49:21.000Z	2019-05-20T19:01:36.000Z	77.22	us-west-2	AVAILABLE

3. In the **Reserve Contact** dialog box, review your contact reservation information.

a. (optional) Under **Tags**, enter a key and value for each tag you want to add.

b. Choose **Reserve**.



Reserve contact

You are about to reserve a contact.

Reservation information

Satellite catalog number 25994	Ground station Ohio 1
Mission profile TERRA (us-west-2)	Max elevation (degrees) 8.17
Start time 2019-05-22T01:48:03.000Z	End time 2019-05-22T01:51:19.000Z

Tags- *optional*

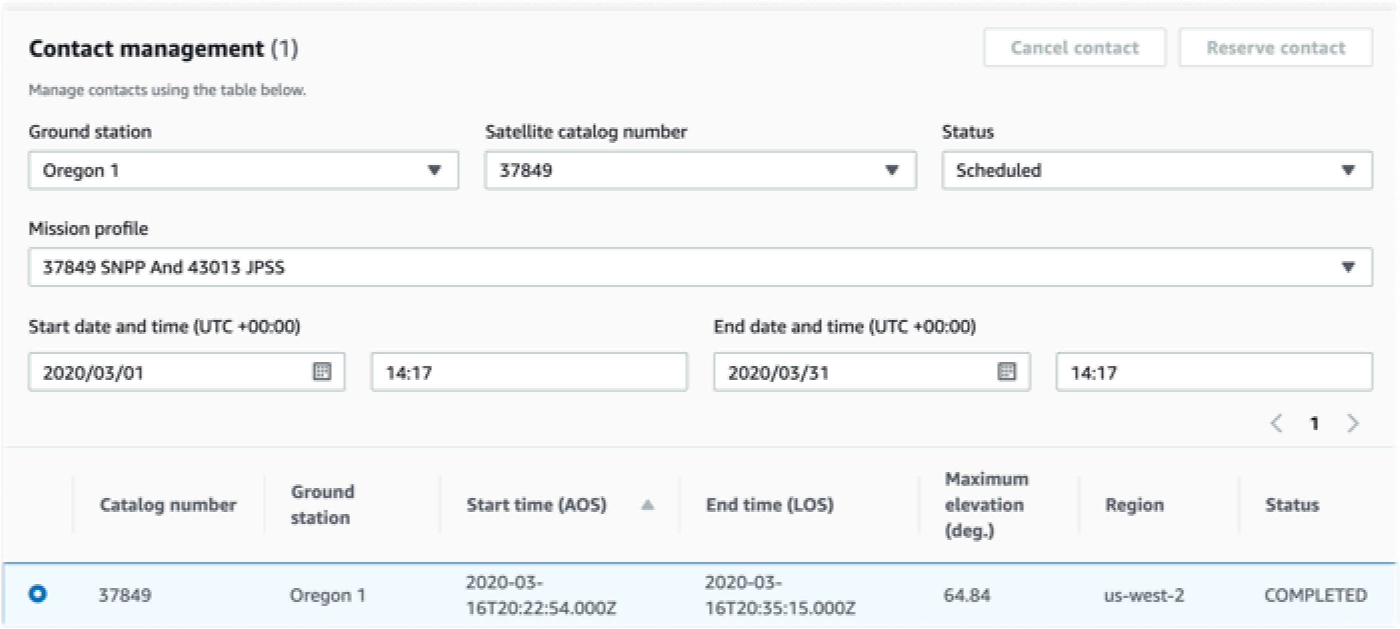
Add optional tags to the contact reservation.

AWS Ground Station will use the configuration data from your mission profile to execute a contact at the specified ground station.

View Scheduled and Completed Contacts

Once you schedule contacts, you can use the AWS Ground Station console to view the details of scheduled and completed contacts.

In the **Contact management** table, choose the parameters you want to use to search for scheduled and completed contacts. Ensure that you are viewing **Scheduled** or **Completed** contacts by using the **Status** filter.



Contact management (1)

Manage contacts using the table below.

Ground station: Oregon 1

Satellite catalog number: 37849

Status: Scheduled

Mission profile: 37849 SNPP And 43013 JPSS

Start date and time (UTC +00:00): 2020/03/01 14:17

End date and time (UTC +00:00): 2020/03/31 14:17

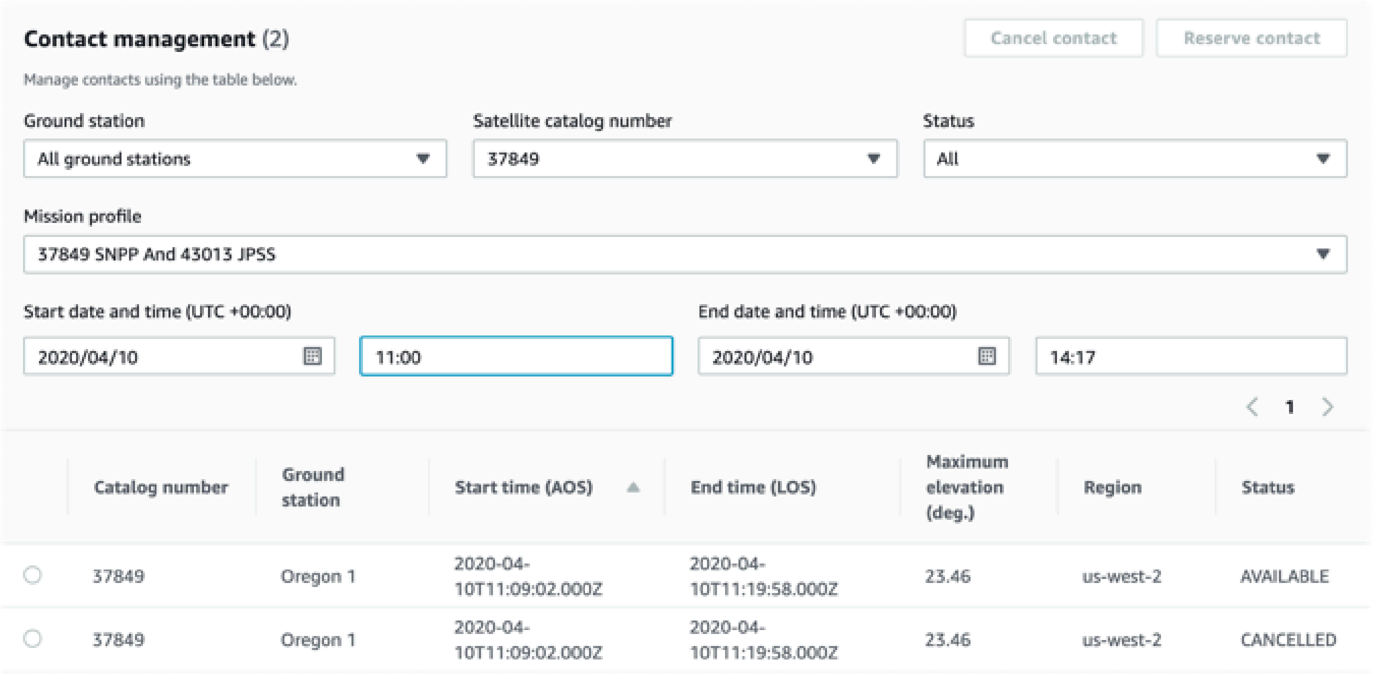
	Catalog number	Ground station	Start time (AOS)	End time (LOS)	Maximum elevation (deg.)	Region	Status
●	37849	Oregon 1	2020-03-16T20:22:54.000Z	2020-03-16T20:35:15.000Z	64.84	us-west-2	COMPLETED

Your scheduled or completed contact(s) will be listed if the contact(s) matches the parameters.

Cancelling Contacts

You can use the AWS Ground Station console to cancel scheduled contacts

1. In the **Contact management** table, choose the parameters you want to use to search for scheduled and completed contacts. Ensure that you are viewing **Scheduled** contacts by using the **Status** filter.
2. Choose the contact you want to cancel in the list of scheduled contacts. Then, choose **Cancel Contact**.
3. In the **Cancel contact** dialog box, choose **Ok**.



Contact management (2)

Manage contacts using the table below.

Ground station: All ground stations

Satellite catalog number: 37849

Status: All

Mission profile: 37849 SNPP And 43013 JPSS

Start date and time (UTC +00:00): 2020/04/10 11:00

End date and time (UTC +00:00): 2020/04/10 14:17

	Catalog number	Ground station	Start time (AOS)	End time (LOS)	Maximum elevation (deg.)	Region	Status
○	37849	Oregon 1	2020-04-10T11:09:02.000Z	2020-04-10T11:19:58.000Z	23.46	us-west-2	AVAILABLE
○	37849	Oregon 1	2020-04-10T11:09:02.000Z	2020-04-10T11:19:58.000Z	23.46	us-west-2	CANCELLED

The contact's status will be *CANCELLED*.

Naming Satellites

The AWS Ground Station console has the ability to display a user defined name for a satellite along with the Norad ID when using the Contacts page. Displaying the satellite name makes it much easier to select the correct satellite when scheduling. To do this, tags can be used.

Tagging AWS Ground Station Satellites can be done via the tag-resource API with the AWS CLI or one of the AWS SDKs. This guide will cover using the AWS Ground Station CLI to tag the public broadcast satellite Aqua (Norad ID 27424) in `us-west-2`.

AWS Ground Station CLI

The AWS CLI can be used to interact with AWS Ground Station. Before using AWS CLI to tag your satellites, the following AWS CLI prerequisites must be fulfilled:

- Ensure that AWS CLI is installed. For information about installing AWS CLI, see Installing the AWS CLI version 2.
- Ensure that AWS CLI is configured. For information about configuring AWS CLI, see Configuring the AWS CLI version 2.

- Save your frequently used configuration settings and credentials in files that are maintained by the AWS CLI. You need these settings and credentials to reserve and manage your AWS Ground Station contacts with AWS CLI. For more information about saving your configuration and credential settings, see Configuration and Credential File Settings .

Once AWS CLI is configured and ready to use, review the AWS Ground Station CLI Command Reference page to familiarize yourself with available commands. Follow the AWS CLI command structure when using this service and prefix your commands with `groundstation` to specify AWS Ground Station as the service you want to use. For more information on the AWS CLI command structure, see Command Structure in the AWS CLI page. An example command structure is provided below.

```
aws groundstation <command> <subcommand> [options and parameters]
```

Name a Satellite

First you need to get the ARN for the satellite(s) you wish to tag. This can be done via the list-satellites API in the AWS CLI:

```
aws groundstation list-satellites --region us-west-2
```

Running the above CLI command will return an output similar to this:

```
{
    "satellites": [
        {
            "groundStations": [
                "Ohio 1",
                "Oregon 1"
            ],
            "noradSatelliteID": 27424,
            "satelliteArn":
 "arn:aws:groundstation::111111111111:satellite/11111111-2222-3333-4444-555555555555",
            "satelliteId": "11111111-2222-3333-4444-555555555555"
        }
    ]
}
```

Find the satellite you wish to tag and note down the `satelliteArn`. One important caveat for tagging is that the tag-resource API requires a regional ARN, and the ARN returned by list-satellites is global. For the next step, you should augment the ARN with the region you would like to see the tag in (likely the region you schedule in). For this example, we are using `us-west-2`. With this change, the ARN will go from:

```
arn:aws:groundstation::111111111111:satellite/11111111-2222-3333-4444-555555555555
```

to:

```
arn:aws:groundstation:us-
west-2:111111111111:satellite/11111111-2222-3333-4444-555555555555
```

In order to show the satellite name in the console, the satellite must have a tag with “`Name`" as the key. Additionally, because we are using the AWS CLI, the quotation marks must be escaped with a backslash. The tag will look something like:

```
{\"Name\":\"AQUA\"}
```

Next, you will call the tag-resource API to tag the satellite. This can be done with the AWS CLI like so:

```
aws groundstation tag-resource --region us-west-2 --resource-arn
 arn:aws:groundstation:us-
west-2:111111111111:satellite/11111111-2222-3333-4444-555555555555 --tags {\"Name\":
\"AQUA\"}
```

After doing this, you'll be able to see the name you set for the satellite in the AWS Ground Station console.

Change the Name For a Satellite

If you want to change the name for a satellite, you can simply call tag-resource with the satellite ARN again with the same “`Name`” key, but with a different value in the tag. This will update the existing tag and show the new name in the console. An example call for this looks like:

```
aws groundstation tag-resource --region us-west-2 --resource-arn
 arn:aws:groundstation:us-
```

```
west-2:111111111111:satellite/11111111-2222-3333-4444-555555555555 --tags {\"Name\":
\"NewName\"}
```

Remove the Name For a Satellite

The name set for a satellite can be removed with the untag-resource API. This API needs the satellite ARN with the region the tag is in, and a list of tag keys. For the name, the tag key is “Name”. An example call to this API using the AWS CLI looks like:

```
aws groundstation untag-resource --region us-west-2 --resource-arn
 arn:aws:groundstation:us-
west-2:111111111111:satellite/11111111-2222-3333-4444-555555555555 --tag-keys Name
```

Reserving and Managing Contacts with AWS CLI

You can use AWS CLI to reserve and manage your contacts in AWS Ground Station. Before using AWS CLI to reserve and manage contacts, the following AWS CLI prerequisites must be fulfilled:

- Ensure that AWS CLI is installed. For information about installing AWS CLI, see Installing the AWS CLI version 2.
- Ensure that AWS CLI is configured. For information about configuring AWS CLI, see Configuring the AWS CLI version 2.
- Save your frequently used configuration settings and credentials in files that are maintained by the AWS CLI. You need these settings and credentials to reserve and manage your AWS Ground Station contacts with AWS CLI. For more information about saving your configuration and credential settings, see Configuration and Credential File Settings .

Once AWS CLI is configured and ready to use, review the AWS Ground Station CLI Command Reference page to familiarize yourself with available commands. Follow the AWS CLI command structure when using this service and prefix your commands with `groundstation` to specify AWS Ground Station as the service you want to use. For more information on the AWS CLI command structure, see Command Structure in the AWS CLI page. An example command structure is provided below.

```
aws groundstation <command> <subcommand> [options and parameters]
```

Use the following topics to reserve, view, and cancel contacts with AWS CLI.

Topics

View and List Contacts with AWS CLI

To list and view `CANCELLED`, `COMPLETED`, or `SCHEDULED` contacts with AWS CLI, run `aws groundstation list-contacts` with the following parameters.

- **Start Time** - Specify the start time of your contact with `--start-time <value>`. The following is an acceptable time value format: `YYYY-MM-DDTHH:MM:SSZ`
- **End Time** - Specify the end time of your contact with `--end-time <value>`. The following is an acceptable time value format: `YYYY-MM-DDTHH:MM:SSZ`
- **Status List** - Specify the status of your contact with `--status-list <value>`. Acceptable values include `AVAILABLE`, `CANCELLED`, `COMPLETED`, or `SCHEDULED`. To see a full list of valid values, see list-contacts.

To list and view `AVAILABLE` contacts with AWS CLI the following parameters are required in addition to the ones listed above.

- **Ground Station ID** - Specify your ground station's ID with `--ground-station <value>`.
- **Mission Profile ARN** - Specify your mission profile's ARN with `--mission-profile-arn <value>`.
- **Satellite ARN** - Specify your satellite ARN with `--satellite-arn <value>`.

You can use `list` commands to look up your resources. For more information on specifying your parameters, see list-contacts

An example command to list available contacts is provided below.

```
aws groundstation --region us-east-2 list-contacts --ground-station 'Ohio 1'
 --mission-profile-arn 'arn:aws:groundstation:us-east-2:123456789012:mission-
profile/11111111-2222-3333-4444-555555555555' --satellite-arn
```

```
 'arn:aws:groundstation::123456789012:satellite/11111111-2222-3333-4444-555555555555'
 --start-time '2020-04-10T00:09:22Z' --end-time '2020-04-10T00:11:22' --status-list
 'AVAILABLE'
```

An example of a list of available contacts is provided below.

```
{
  "contactList": [
    {
      "contactStatus": "AVAILABLE",
      "endTime": "2020-04-15T03:16:35-06:00",
      "groundStation": "Oregon 1",
      "maximumElevation": {
        "unit": "DEGREE_ANGLE",
        "value": 11.22
      },
      "missionProfileArn": "arn:aws:groundstation:us-west-2:111111111111:mission-
profile/11111111-2222-3333-4444-555555555555",
      "region": "us-west-2",
      "satelliteArn":
 "arn:aws:groundstation::111111111111:satellite/11111111-2222-3333-4444-555555555555",
      "startTime": "2020-04-15T03:06:08-06:00"
    }
  ]
}
```

Reserve a Contact with AWS CLI

AWS CLI gives you the option to reserve contacts by the minute. This feature is unique to the AWS CLI and cannot be done in the AWS Ground Station console.

To reserve contacts with AWS CLI, run `aws groundstation reserve-contact` with the following parameters.

- **Ground Station ID** - Specify your ground station's ID with `--ground-station <value>`.
- **Mission Profile ARN** - Specify your mission profile's ARN with `--mission-profile-arn <value>`.
- **Satellite ARN** - Specify your satellite ARN with `--satellite-arn <value>`.
- **Start Time** - Specify the start time of your contact with `--start-time <value>`. The following is an acceptable time value format: `YYYY-MM-DDTHH:MM:SSZ`

- **End Time** - Specify the end time of your contact with `--end-time <value>`. The following is an acceptable time value format: `YYYY-MM-DDTHH:MM:SSZ`

Contact reservation is an asynchronous process. The response to the `reserve-contact` command provides the contact identifier. In order to determine the outcome of the asynchronous reservation process, use `describe-contact`. For more information on this, see the section below titled Describe a Contact with AWS CLI.

You can use `list` commands to look up your resources. For more information on specifying your parameters, see reserve-contact.

An example command of reserving a contact is provided below.

```
aws groundstation reserve-contact --ground-station 'Ohio 1' --mission-
profile-arn 'arn:aws:groundstation:us-east-2:123456789012:mission-
profile/11111111-2222-3333-4444-555555555555' --satellite-arn
 'arn:aws:groundstation::123456789012:satellite/11111111-2222-3333-4444-555555555555'
 --start-time '2020-04-10T00:09:22Z' --end-time '2020-04-10T00:11:22'
```

An example of a successfully reserved contact is provided below.

```
{
  "contactId": "11111111-2222-3333-4444-555555555555"
}
```

Describe a Contact with AWS CLI

To see the status of a contact/reservation with AWS CLI, use the `describe-contact` CLI command. This is helpful for verifying the outcome of the asynchronous contact reservation process, monitoring the status of an in-progress contact, and determining the status of a finished contact.

To describe contacts with AWS CLI, run `aws groundstation describe-contact` with the following parameters.

- **Contact ID** - Specify your contact ID with `--contact-id <value>`.

You can use `list` commands to look up your resources. For more information on specifying your parameters, see describe-contact.

An example command of describing a contact is provided below.

```
aws groundstation describe-contact --contact-id 11111111-2222-3333-4444-555555555555
```

An example of a successfully scheduled contact is provided below.

```
{
  "groundStation": "Ireland 1",
  "tags": {},
  "missionProfileArn": "arn:aws:groundstation:us-west-2:111111111111:mission-
profile/11111111-2222-3333-4444-555555555555",
  "region": "us-west-2",
  "contactId": "11111111-2222-3333-4444-555555555555",
  "prePassStartTime": 1645850471.0,
  "postPassEndTime": 1645851172.0,
  "startTime": 1645850591.0,
  "maximumElevation": {
      "value": 12.66,
      "unit": "DEGREE_ANGLE"
  },
  "satelliteArn":
 "arn:aws:groundstation::111111111111:satellite/11111111-2222-3333-4444-555555555555",
  "endTime": 1645851052.0,
  "contactStatus": "SCHEDULED"
}
```

Cancel a Contact with AWS CLI

To cancel a contact with AWS CLI, run `aws groundstation cancel-contact` with the following parameters.

- **Region** - Specify your ground station's region with `--region <value>`.
- **Contact ID** - Specify the contact ID with `--contact-id <value>`.

You can use `list` commands to look up your resources. For more information on specifying your parameters, see cancel-contacts

An example command of reserving a contact is provided below.

```
aws groundstation --region us-east-2 cancel-contact --contact-id
 '11111111-2222-3333-4444-555555555555'
```

An example of a successfully cancelled contact is provided below.

```
{
  "contactId": "11111111-2222-3333-4444-555555555555"
}
```

Data Delivery to Amazon EC2

AWS Ground Station delivers your contact data asynchronously to an Amazon Simple Storage Service (Amazon S3) bucket in your account or synchronously by streaming it to and from an Amazon Elastic Compute Cloud (Amazon EC2) instance in your account. The following steps describe how to configure the resources required to stream contact data to and from an Amazon EC2 instance. See the Getting Started with AWS Ground Station guide for information about data delivery to Amazon S3.

Topics

Step 1: 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. Use the steps below.

1. In your AWS Management Console, choose an AWS Region in which you plan to reserve contacts. You need to create a key pair for every AWS Region you choose.

> **Note**
>
> AWS Ground Station is not yet available for all regions. Ensure that AWS Ground Station is supported by your desired AWS Region. For more information about AWS Ground Station antenna locations, see AWS Ground Station FAQs.

2. Follow the guide Create Key Pairs in the *Amazon EC2 User Guide* to create the key pairs.
3. Repeat for other AWS Regions if needed.

Step 2: 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 Linux Bastion Hosts on AWS.

For your convenience, instructions to quickly add a bastion host to your Linux environment in AWS are below. While this is not required, it is recommended best practice.

1. Login to your AWS account.
2. In the Linux Bastion Hosts on the AWS Cloud: Quick Start Reference Deployment page, choose **Launch Quick Start (for new VPC)**.
3. In the **Create Stack** page, choose **Next**. The template is pre-populated.
4. In the **Specify stack** details page, make edits and changes in the following boxes:

    a. Enter a stack name for your host in the **Stack Name** box.

    b. For **Availability Zones**, select the Availability Zones you wish to use for the subnets in the VPC. At least two Availability Zones must be selected.

    c. For **Allowed bastion external access CIDR**, enter the CIDR block that you would like to enable SSH access from. If you are unsure, you can use the value of **0.0.0.0/0** to enable SSH access from any host that has the SSH key.

    d. For **Key pair name**, choose the key pair name you created in the section called "Step 1: Create EC2 SSH Key Pair".

    e. For **Bastion instance type**, choose **t2.micro**.

    > ⚠ **Important**
    >
    > The **t2.micro** instance type is not available for the Europe (Stockholm) Region (eu-north-1). If you are using AWS Ground Station in the Europe (Stockholm) Region (eu-north-1), choose **t3.micro**.

    f. For **TCP forwarding**, choose **true**.

    g. (Optional) Make other edits and changes as necessary. To customize your deployment, you can change your VPC configuration, choose the number and type of bastion host instances, enable TCP or X11 forwarding, and enable a default or custom banner for your bastion hosts.

   h. Choose **Next**.

5. In the **Configure stack options** page, make any changes or edits as necessary.
6. Choose **Next**.
7. Review the details of your bastion host and select the two **Capabilities** acknowledgements. Then, choose **Create stack**.

Step 3: Choose and Customize an AWS CloudFormation Template

Today, you can configure multiple streams of data per contact to flow into your VPC. These data streams are available in two different formats. Data streams containing VITA-49 Signal/IP data can be configured for S-Band and X-Band signals up to 54 MHz in bandwidth. VITA-49 Extension data/IPs can be configured for demodulated and/or decoded X-Band signals up to 500 MHz in bandwidth.

After you onboard 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 preconfigured AWS CloudFormation templates that use public broadcast satellites. These templates make it easy for you to start using AWS Ground Station. For more information about AWS CloudFormation, see What is AWS CloudFormation?

It is important to note that you need to have data processing software or data storage software listening to the localhost side of Data Defender of the Amazon EC2 instance. This software is what you will use to store and/or process the data delivered to the Amazon EC2 instance during a contact.

Configuring your Amazon EC2 Instance Settings

The AWS CloudFormation templates provided in this section are configured to use Amazon EC2 m5.4xlarge instance types by default. However, we encourage you to customize and choose the right Amazon EC2 instance settings for your use case. Requirements such as storage I/O and CPU performance should be considered when choosing your instance settings. For example, running a software modem on a receiver instance may require compute-optimized instances with more cores and a higher clock speed. The best way to determine the right instance settings for your use case is to test your instance settings with your workload, and Amazon EC2 makes it easy to switch between instance settings. Use the templates and customize the instance settings for your needs.

As a general recommendation, AWS Ground Station encourages the use of instances that support enhanced networking for your uplinks and downlinks, such as AWS Nitro System. For more information about enhanced networking, see Enabling enhanced networking with the Elastic Network Adapter (ENA) on Linux instances.

In addition to configuring Amazon EC2 instance types, the AWS CloudFormation templates configure the base Amazon Machine Images (AMI) to be used for the instance. The AWS Ground Station base contains the software needed to receive data from the service pre-installed on your EC2 instance. For more information about AMIs, see Amazon Machine Images (AMI).

Manually Creating and Configuring Resources

The sample AWS CloudFormation templates in this section configure all the resources necessary to begin executing satellite contacts. If you prefer to manually create and configure the resources required to begin executing satellite contacts, you will need to do the following:

- Create AWS Ground Station configs. For more information about manually creating AWS Ground Station configs, see Create Config AWS CLI Command Reference or Create Config API Reference.
- Create an AWS Ground Station mission profile. For more information about manually creating an AWS Ground Station mission profile, see Create Mission Profile AWS CLI Command Reference or Create Mission Profile API Reference.
- Create an AWS Ground Station dataflow endpoint group. For more information about manually creating an AWS Ground Station dataflow endpoint group, see Create Dataflow Endpoint Group AWS CLI Command Reference or Create Dataflow Endpoint Group API Reference.
- Create an EC2 instance. For more information about manually creating an EC2 instance for use with AWS Ground Station, see Create an Amazon EC2 Instance.
- Configure your EC2 instance's security group settings to allow AWS Ground Station to send data to/from your EC2 instance. For more information about manually configuring your EC2 instance's security group settings, see Create Security Group AWS CLI Command Reference or Create Security Group API Reference.

Choose a Template

AWS Ground Station provides templates that demonstrate how to use the service and can be accessed in different ways. Use this guide to find the right template for you.

Using a preconfigured template

You can use a preconfigured template to receive direct broadcast data from the Aqua, SNPP, JPSS-1/NOAA-20, and Terra satellites. These templates contain the required AWS CloudFormation resources to schedule and execute contacts. The `AquaSnppJpss` template comprises the necessary AWS CloudFormation resources to receive demodulated and decoded direct broadcast data. Use this template as a starting point if you plan to process the data using NASA Direct Readout Labs software (RT-STPS and IPOPP). The `AquaSnppJpssTerraDigIF` template comprises the necessary AWS CloudFormation resources to receive raw digitized intermediate frequency (DigIF) direct broadcast data. Use this template as a starting point for processing the data using a software defined radio (SDR). The `DirectBroadcastSatelliteWbDigIfEc2DataDelivery` template comprises the necessary AWS CloudFormation resources to receive raw Wideband digitized intermediate frequency (DigIF) direct broadcast data via the AWS Ground Station Agent.

Narrowband Data Delivery Templates:

- the section called "AquaSnppJpss Template (Narrowband)"
- the section called "AquaSnppJpssTerraDigIF Template (Narrowband)"

Wideband DigIF Data Delivery Templates:

- the section called "Direct Broadcast Satellite Wideband DigIF Template (Wideband)"

> ⚠ **Important**
>
> Satellites must be onboarded to the service in order to access AMIs with the AWS CloudFormation templates.

Using your own satellites

Configuring your own satellites requires a different set of parameters and resources. This is difficult to do on your own. The AWS Ground Station team is available to help you configure your own satellites for use and can help you configure resources for downlink, uplink, and uplink echo streams. To configure your own satellite to use with AWS Ground Station, contact AWS Support.

Accessing Templates

You can access the templates in the regional Amazon S3 bucket below. Note that the following link uses a regional S3 endpoint. Change `<us-west-2>` to the region in which you are creating the AWS CloudFormation stack.

```
s3://groundstation-cloudformation-templates-us-west-2/
```

You can also download the templates using the AWS CLI. For information on configuring the AWS CLI, see Configuring the AWS CLI.

AquaSnppJpss Template (Narrowband)

The AWS CloudFormation template named `AquaSnppJpss.yml` is designed to give you quick access to start receiving data for the Aqua, SNPP, and JPSS-1/NOAA-20 satellites. It contains an Amazon EC2 instance and the required AWS Ground Station resources to schedule contacts and receive demodulated and decoded direct broadcast data. This template is a good starting point if you plan to process the data using NASA Direct Readout Labs software (RT-STPS and IPOPP).

If Aqua, SNPP, and JPSS-1/NOAA-20 are not onboarded to your account, see Customer Onboarding.

Important

The Amazon EC2 instance needs to be stopped before applying the template. Check to ensure that the instance is stopped until you are ready to use it.

You can access the template by accessing the customer onboarding S3 bucket. Note that the links below use a regional S3 bucket. Change `<us-west-2>` to the region in which you are creating the AWS CloudFormation stack.

Note

The following instructions use YAML. However, the templates are available in both YAML and JSON format. To use JSON, replace `<.yml>` with `<.json>`.

To download the template using AWS CLI, use the following command:

```
aws s3 cp s3://groundstation-cloudformation-templates-us-west-2/AquaSnppJpss.yml .
```

You can view and download the template in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-cloudformation-templates-us-
west-2/AquaSnppJpss.yml
```

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

```
https://groundstation-cloudformation-templates-us-west-2.s3.us-west-2.amazonaws.com/
AquaSnppJpss.yml
```

What resources does the template define?

The `AquaSnppJpss` template includes the following resources:

- **Data Delivery Service Role** - AWS Ground Station assumes this role to create/delete ENIs in your account in order to stream data.
- (Optional) **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 launched.
- **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. You can modify this in a way that limits traffic to a specific set of IP addresses.
- **Receiver Instance Network Interface** - An elastic network interface that provides a fixed IP address for AWS Ground Station to connect to. This attaches to the receiver instance on `eth1`.
- **Receiver Instance Interface Attachment** - An elastic network interface that attaches to your Amazon EC2 instance.

- (Optional) **CloudWatch Event Triggers** - AWS Lambda Function that is triggered using CloudWatch Events sent by AWS Ground Station before and after a contact. The AWS Lambda Function will start and optionally stop your Receiver Instance.
- (Optional) **EC2 Verification for Contacts** - The option to use Lambda to set up a verification system of your Amazon EC2 instance(s) for contacts with SNS notification. It is important to note that this may incur charges depending on your current usage.
- **Dataflow Endpoint Group** - The AWS Ground Station dataflow endpoint group 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.
- **Tracking Config** - The AWS Ground Station tracking config defines how the antenna system tracks your satellite as it moves through the sky.
- **Ground Station Amazon Machine Image Retrieval Lambda** - The option to select what software is installed in your instance and the AMI of your choice. The software options include `DDX 2.6.2 Only` and `DDX 2.6.2 with qRadio 3.6.0`. If you want to use Wideband DigIF Data Delivery and the AWS Ground Station Agent, please use the AquaSnppJpssTerraDigIF Template (Narrowband). These options will continue to expand as additional software updates and features are released.

In addition, the template provides the following resources for the Aqua, SNPP, JPSS-1/NOAA-20 satellites:

- A downlink demod/decode config for JPSS-1/NOAA-20 and SNPP, and a downlink demod/ decode config for Aqua.
- A mission profile for JPSS-1/NOAA-20 and SNPP, and a mission profile for Aqua.

The values and parameters for the satellites in this template are already populated. These parameters make it easy for you to use AWS Ground Station immediately with these satellites. You do not need to configure your own values in order to use AWS Ground Station when using this template. However, you can customize the values to make the template work for your use case.

Where do I receive my data?

The dataflow endpoint group is set up to use the receiver instance network interface that part of the template creates. The receiver instance uses Data Defender to receive the data stream from AWS Ground Station on the port defined by the dataflow endpoint. Once received, the data is available for consumption via UDP port 50000 on the loopback adapter of the receiver instance. For more information about setting up a dataflow endpoint group, see AWS::GroundStation::DataflowEndpointGroup.

AquaSnppJpssTerraDigIF Template (Narrowband)

The AWS CloudFormation template named `AquaSnppJpssTerraDigIF.yml` is designed to give you quick access to start receiving digitized intermediate frequency (DigIF) data for the Aqua, SNPP, JPSS-1/NOAA-20, and Terra satellites. It contains an Amazon EC2 instance and the required AWS CloudFormation resources to receive raw DigIF direct broadcast data. This template is a good starting point for processing the data using a software defined radio (SDR).

If Aqua, SNPP, JPSS-1/NOAA-20, and Terra are not onboarded to your account, see Customer Onboarding.

⚠ **Important**

The Amazon EC2 instance needs to be stopped before applying the template. Check to ensure that the instance is stopped until you are ready to use it.

You can access the template by accessing the customer onboarding S3 bucket. Note that the links below use a regional S3 bucket. Change `<us-west-2>` to the region in which you are creating the AWS CloudFormation stack.

ⓘ **Note**

The following instructions use YAML. However, the templates are available in both YAML and JSON format. To use JSON, replace `<.yml>` with `<.json>`.

To download the template using AWS CLI, use the following command:

```
aws s3 cp s3://groundstation-cloudformation-templates-us-west-2/
AquaSnppJpssTerraDigIF.yml .
```

You can view and download the template in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-cloudformation-templates-us-
west-2/AquaSnppJpssTerraDigIF.yml
```

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

```
https://groundstation-cloudformation-templates-us-west-2.s3.us-west-2.amazonaws.com/
AquaSnppJpssTerraDigIF.yml
```

What resources does the template define?

The `AquaSnppJpssTerraDigIF` template includes the following resources:

- **Data Delivery Service Role** - AWS Ground Station assumes this role to create/delete ENIs in your account in order to stream data.
- (Optional) **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 launched.
- **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. You can modify this in a way that limits traffic to a specific set of IP addresses.
- **Receiver Instance Network Interface** - An elastic network interface that provides a fixed IP address for AWS Ground Station to connect to. This attaches to the receiver instance on `eth1`.
- **Receiver Instance Interface Attachment** - An elastic network interface that attaches to your Amazon EC2 instance.
- (Optional) **CloudWatch Event Triggers** - AWS Lambda Function that is triggered using CloudWatch Events sent by AWS Ground Station before and after a contact. The AWS Lambda Function will start and optionally stop your Receiver Instance.
- (Optional) **EC2 Verification for Contacts** - The option to use Lambda to set up a verification system of your Amazon EC2 instance(s) for contacts with SNS notification. It is important to note that this may incur charges depending on your current usage.

- **Dataflow Endpoint Group** - The AWS Ground Station dataflow endpoint group 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.
- **Tracking Config** - The AWS Ground Station tracking config defines how the antenna system tracks your satellite as it moves through the sky.
- **Downlink Dig IF Endpoint Config** - A defined endpoint used to downlink data from your satellite.
- **Ground Station Amazon Machine Image Retrieval Lambda** - The option to select what software is installed in your instance and the AMI of your choice. The software options include `DDX 2.6.2 Only` and `DDX 2.6.2 with qRadio 3.6.0`. These options will continue to expand as additional software updates and features are released.

In addition, the template provides the following resources for the Aqua, SNPP, JPSS-1/NOAA-20, and Terra satellites:

- A downlink DigIF antenna config for Aqua, SNPP, JPSS-1/NOAA-20, and Terra.
- A mission profile for JPSS-1/NOAA-20 and SNPP, a mission profile for Aqua, and a mission profile for Terra.

The values and parameters for the satellites in this template are already populated. These parameters make it easy for you to use AWS Ground Station immediately with these satellites. You do not need to configure your own values in order to use AWS Ground Station when using this template. However, you can customize the values to make the template work for your use case.

Where do I receive my data?

The dataflow endpoint group is set up to use the receiver instance network interface that part of the template creates. The receiver instance uses Data Defender to receive the data stream from AWS Ground Station on the port defined by the dataflow endpoint. Once received, the data is available for consumption via UDP port 50000 on the loopback adapter of the receiver instance. For more information about setting up a dataflow endpoint group, see AWS::GroundStation::DataflowEndpointGroup.

Direct Broadcast Satellite Wideband DigIF Template (Wideband)

The AWS CloudFormation template named `DirectBroadcastSatelliteWbDigIfEc2DataDelivery.yml` is designed to give you quick access to start receiving digitized intermediate frequency (DigIF) data for the Aqua, SNPP, JPSS-1/NOAA-20, and Terra satellites. It contains an Amazon EC2 instance and the required AWS CloudFormation resources to receive raw DigIF direct broadcast data. This template is a good starting point for processing the data using a software defined radio (SDR).

If Aqua, SNPP, JPSS-1/NOAA-20, and Terra are not onboarded to your account, see Customer Onboarding.

> ⚠ **Important**
>
> The Amazon EC2 instance needs to be stopped before applying the template. Check to ensure that the instance is stopped until you are ready to use it.

You can access the template by accessing the customer onboarding S3 bucket. Note that the links below use a regional S3 bucket. Change `<us-west-2>` to the region in which you are creating the AWS CloudFormation stack.

> ⓘ **Note**
>
> The following instructions use YAML. However, the templates are available in both YAML and JSON format. To use JSON, replace `<.yml>` with `<.json>`.

To download the template using AWS CLI, use the following command:

```
aws s3 cp s3://groundstation-cloudformation-templates-us-west-2/agent/ec2_delivery/
DirectBroadcastSatelliteWbDigIfEc2DataDelivery.yml .
```

You can view and download the template in the console by navigating to the following URL in your browser:

```
https://s3.console.aws.amazon.com/s3/object/groundstation-cloudformation-templates-us-
west-2/agent/ec2_delivery/DirectBroadcastSatelliteWbDigIfEc2DataDelivery.yml
```

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

```
https://groundstation-cloudformation-templates-us-west-2.s3.us-west-2.amazonaws.com/
agent/ec2_delivery/DirectBroadcastSatelliteWbDigIfEc2DataDelivery.yml
```

What resources does the template define?

The `DirectBroadcastSatelliteWbDigIfEc2DataDelivery` template includes the following resources:

- (Optional) **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 launched.
- **Data Delivery Key** - AWS KMS Key used to encrypt dataflows.
- **Ground Station Key Role** - The IAM role that AWS Ground Station will assume to access and use the AWS KMS Key to decrypt dataflows
- **Ground Station Key Access Policy** - The IAM policy defining the actions AWS Ground Station can take on the Data Delivery Key
- **Receiver Instance Elastic Network Interface** - (Conditional) An elastic network interface is created in the subnet specified by **PublicSubnetId** if provided. This is required if the receiver instance is in a private subnet. The elastic network interface will be associated with the EIP and attached to the receiver instance.
- **Receiver Instance Elastic IP** - An elastic IP that AWS Ground Station will connect to. This attaches to the receiver instance or elastic network interface.
- One of the following Elastic IP associations:
  - **Receiver Instance to Elastic IP Association** - The association of the Elastic IP to your receiver instance, if **PublicSubnetId** is not specified. This requires that **SubnetId** reference a public subnet.
  - **Receiver Instance Elastic Network Interface to Elastic IP Association** - The association of the elastic IP to the receiver instance elastic network interface, if **PublicSubnetId** is specified.

- (Optional) **CloudWatch Event Triggers** - AWS Lambda Function that is triggered using CloudWatch Events sent by AWS Ground Station before and after a contact. The AWS Lambda Function will start and optionally stop your Receiver Instance.
- (Optional) **EC2 Verification for Contacts** - The option to use Lambda to set up a verification system of your Amazon EC2 instance(s) for contacts with SNS notification. It is important to note that this may incur charges depending on your current usage.
- **Dataflow Endpoint Group** - The AWS Ground Station dataflow endpoint group that defines the endpoints used to send/receive data to/from your satellite.
- **Tracking Config** - The AWS Ground Station tracking config defines how the antenna system tracks your satellite as it moves through the sky.

In addition, the template provides the following resources for the Aqua, SNPP, JPSS-1/NOAA-20, and Terra satellites:

- A downlink config for JPSS-1/NOAA-20 and SNPP, a downlink config for Aqua, and a downlink config for Terra.
- A mission profile for JPSS-1/NOAA-20 and SNPP, a mission profile for Aqua, and a mission profile for Terra.

The values and parameters for the satellites in this template are already populated. These parameters make it easy for you to use AWS Ground Station immediately with these satellites. You do not need to configure your own values in order to use AWS Ground Station when using this template. However, you can customize the values to make the template work for your use case.

Where do I receive my data?

The dataflow endpoint group is set up to use the receiver instance network interface that part of the template creates. The receiver instance uses the AWS Ground Station Agent to receive the data stream from AWS Ground Station on the port defined by the dataflow endpoint. For more information about setting up a dataflow endpoint group, see AWS::GroundStation::DataflowEndpointGroup. For more information about the AWS Ground Station Agent, see AWS Ground Station Agent User Guide.

Create an Amazon EC2 Instance

Note

It is not necessary nor recommended to create your resources for AWS Ground Station (including Amazon EC2 instances) manually as AWS Ground Station provides premade AWS CloudFormation templates for this (See Step 3: Choose and Customize an AWS CloudFormation Template for more information). If using AWS CloudFormation templates will not work for your use case, please continue reading.

AWS Ground Station provides Amazon EC2 AMIs that comes pre-loaded with the software needed to take data delivery on an Amazon EC2 instance for either Narrowband or Wideband DigIf Data Delivery.

Important

Satellites must be onboarded to the service in order to access the AWS Ground Station AMIs.

Amazon EC2 AMI with DataDefender

This AMI comes pre-installed with the DataDefender software and it is used for Narrowband data delivery downlink contacts.

The naming scheme for this AMI is `groundstation-al2-ddx$DDX_VERSION-ami-$DATE_PUBLISHED`. A new DDX AMI is published shortly after a new AL2 Amazon EC2 AMI is published. If AWS Ground Station decides to support a new version of the DataDefender software, a new AMI will be published using the updated version.

Selecting an AWS Ground Station AMI with DataDefender

You can access the AWS Ground Station AMI through the **AMIs** tab in the Amazon EC2 console. Once on that page, the AMIs are accessible under the **Private Images** filter.

We recommend sorting the AMIs by the published date and using the most recently published AMI named `groundstation-al2-ddx$DDX_VERSION-ami-$DATE_PUBLISHED`.

Amazon EC2 AMI with the AWS Ground Station Agent

This AMI comes pre-installed with the AWS Ground Station Agent and is used for Wideband DigIF downlink contacts.

The naming scheme for this AMI is `groundstation-al2-gs-agent-ami-*` where * is the date the AMI was built. A new AWS Ground Station Agent AMI is published shortly after a new AL2 Amazon EC2 AMI is published or when a new version of the AWS Ground Station Agent RPM is released.

For more information about the AWS Ground Station Agent see AWS Ground Station Agent User Guide.

Selecting an AWS Ground Station Agent AMI

You can access the AWS Ground Station Agent AMI through the **AMIs** tab in the Amazon EC2 console. Once on that page, the AMIs are accessible under the **Public Images** filter.

We recommend sorting the AMIs by the published date and using the most recently published AMI named `groundstation-al2-gs-agent-ami-$DATE_PUBLISHED`.

Step 4: Configure an AWS CloudFormation Stack

After choosing the template that best applies to your use case, configure an AWS CloudFormation stack. The resources that are created in this procedure are configured to the region that you are in when you create them. This includes the mission profile and its properties that determine to which region your data is delivered.

1. In the AWS Management Console, choose **Services > CloudFormation**.
2. In the navigation pane, choose **Stacks**. Then, choose **Create stack > With new resources (standard)**.
3. In the **Create Stack** page, specify the template that you selected in the section called "Choose a Template" by doing one of the following.

    a. Select **Amazon S3 URL** as your template source, and copy and paste the URL of the template you want to use in **Amazon S3 URL**. Then, choose **Next**.

    b. Select **Upload a template file** as your template source and choose **Choose File**. Upload the template you downloaded in the section called "Choose a Template". Then, choose **Next**.

4. In the **Specify stack details** page, make the following changes:

    a. Enter a name in the **Stack Name** box. We recommend using a simple name to reduce the possibility of errors in the future.

    b. For **CloudWatchEventActions**, choose which actions to perform for the CloudWatch event triggers before and after a contact.

    c. For **CreateEC2VerificationForContacts**, choose whether or not you want to set up a verification system (utilizing Lambda) of your EC2 instance(s) for contacts with SNS notification. It is important to note that this may incur charges depending on your current usage.

    d. For **CreateReceiverInstance**, choose whether or not you want to create an Amazon EC2 receiver instance.

    e. Choose the SSH Key you created in the section called "Step 1: Create EC2 SSH Key Pair".

    f. Choose the **SubnetId** in which you wish to create your Amazon EC2 instance.

        If using the AWS Ground Station Agent a public subnet is required, either for placement of the instance or an elastic network interface; If you specify a private subnet in **SubnetId** in which to place your instance you must also specify a public subnet in **PublicSubnetId** (see below) to use with the AWS Ground Station Agent.

        For non-Agent use-cases we recommend placing your Amazon EC2 instance in a private subnet as a best practice, though it is not required. You can use the Linux Bastion Hosts on the AWS Cloud: Quick Start Reference Deployment to automatically create a private subnet if you have not already configured your account with one in the section called "Step 2: Set Up Your VPC".

        > **Note**
        >
        > Your organization may have another subnet dedicated for your Amazon EC2 instance.

    g. (Optional) Choose the **PublicSubnetId** to use only if using the AWS Ground Station Agent with an instance in a private subnet. This is required if you specified a private subnet in **SubnetId**.

        This subnet must be in your account in the same availability zone as the one specified by **SubnetId**. Providing a **PublicSubnetId** will result in the creation of an elastic network interface in the provided public subnet, attached to your instance. This interface is used for AWS Ground Station Agent network access from your instance which is placed in the private subnet specified in **SubnetId**.

   h. Choose the VPC Stack you created in the section called “Step 2: Set Up Your VPC”.

   i. Choose **Next**.

5. Configure stack options and advanced options for your Amazon EC2 instance.

   a. Add any tags and permissions in the **Tags** and **Permissions** sections.

   b. Make any changes for your **Stack policy**, **Rollback configuration**, **Notification options**, and **Stack creation options**.

   c. Choose **Next**.

6. After reviewing your stack details, select the **Capabilities** acknowledgement, and choose **Create stack**.

Step 5: 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 radio (SDR) installed by default. You need to install an FE processor or SDR 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 SDR depends on which FE processor or SDR you are using. Installation of an FE processor or SDR is beyond the scope of this user guide.

To install and configure FE processor/radio, contact AWS Support.

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

Next Steps

Your AWS Ground Station account and resources are now configured and ready for use. These resources are available to use in the AWS Ground Station console where you can enter satellite data, identify antenna locations, communicate, and schedule antenna time for selected satellites. You can also begin using different tools to monitor activity and configure alarms.

Use the following topics for more information:

- *Listing and Reserving Contacts*
- *Monitoring AWS Ground Station*

Using Cross-Region Data Delivery Service

The AWS Ground Station cross-region data delivery feature gives you the flexibility to send your data from an antenna to an Amazon EC2 instance in your AWS Region. Cross-region data delivery is currently available in all AWS Ground Station supported regions when receiving your contact data in an Amazon S3 Bucket. It is only available in the following antenna-to-destination regions when utilizing data delivery to Amazon EC2:

- US East (Ohio) Region (us-east-2) to US West (Oregon) Region (us-west-2)
- US West (Oregon) Region (us-west-2) to US East (Ohio) Region (us-east-2)

To use cross-region data delivery, you should have an AWS CloudFormation template configured. For more information about choosing and customizing AWS CloudFormation templates, see Step 3: Choose and Customize an AWS CloudFormation Template.

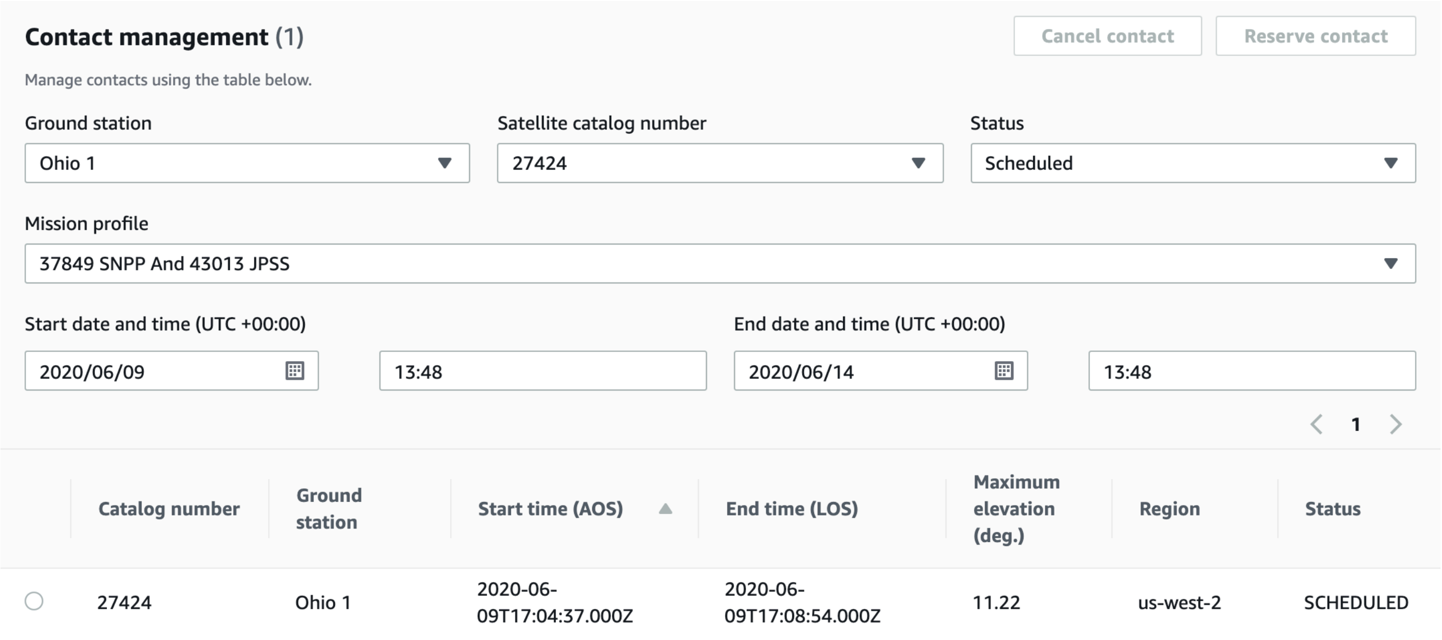
Use the following topics to use cross-region data delivery in AWS Ground Station.

Topics

To use cross-region data delivery in the console

When you reserve a contact in the AWS Ground Station console, choose the mission profile that is configured to deliver the contact data to your desired region. Ensure that all of your parameters are correct and choose **Reserve contact**. If you do not see the desired mission profile in the console, check to make sure you created the mission profile in the region where you are viewing the console.

After reserving your contact, you can view scheduled contacts to verify that you have scheduled cross-region data delivery by viewing the location of the ground station antenna and the destination region. The following image shows a contact that is scheduled for cross-region data delivery. The contact is configured to use the Ohio ground station antennas and deliver data to Oregon.



Contact management (1)

Manage contacts using the table below.

Ground station: Ohio 1

Satellite catalog number: 27424

Status: Scheduled

Mission profile: 37849 SNPP And 43013 JPSS

Start date and time (UTC +00:00): 2020/06/09 13:48

End date and time (UTC +00:00): 2020/06/14 13:48

	Catalog number	Ground station	Start time (AOS)	End time (LOS)	Maximum elevation (deg.)	Region	Status
○	27424	Ohio 1	2020-06-09T17:04:37.000Z	2020-06-09T17:08:54.000Z	11.22	us-west-2	SCHEDULED

To use cross-region data delivery with AWS CLI

When you reserve a contact in AWS CLI, choose the mission profile that is configured to deliver the contact data to your desired region. Specify the desired mission profile's ARN with `--mission-profile-arn <value>`. Ensure that all of your parameters are correct and run the command. If you do not see the desired mission profile ARN when viewing and listing contacts, check to make sure you created the mission profile in the region where you are running AWS CLI.

After reserving your contact, you can view scheduled contacts to verify that you have scheduled cross-region data delivery by viewing the location of the ground station antenna and the destination region. The following output shows a contact that is scheduled for cross-region data delivery. The contact is configured to use the Ohio ground station antennas and deliver the data to Oregon.

```
{
  "contactList": [
    {
      "contactId": "11111111-2222-3333-4444-555555555555",
      "contactStatus": "SCHEDULED",
      "endTime": "2020-05-05T03:16:35-06:00",
      "groundStation": "Ohio 1",
      "maximumElevation": {
        "unit": "DEGREE_ANGLE",
        "value": 26.74
      },
      "missionProfileArn": "arn:aws:groundstation:us-west-2:123456789012:mission-
profile/11111111-2222-3333-4444-555555555555",
      "postPassEndTime": "2020-05-05T03:17:35-06:00",
      "prePassStartTime": "2020-05-05T03:04:08-06:00",
      "region": "us-west-2",
      "satelliteArn":
 "arn:aws:groundstation::123456789012:satellite/11111111-2222-3333-4444-555555555555",
      "startTime": "2020-05-05T03:06:08-06:00"
    }
  ]
}
```

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 about Amazon CloudWatch Events, see the Amazon CloudWatch Events User Guide.
- *AWS EventBridge Events* delivers a near real-time stream of system events that describe changes in AWS resources. EventBridge 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 about EventBridge Events, see the Amazon EventBridge 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 about AWS CloudTrail, see the AWS CloudTrail User Guide.
- *Amazon CloudWatch Metrics* captures metrics for your scheduled contacts when using AWS Ground Station. CloudWatch Metrics enables you to analyze data based on your channel, polarization, and satellite ID to identify signal strength and errors in your contacts. For more information, see Using Amazon CloudWatch Metrics.
- *AWS User Notifications* can be used to set up delivery channels to get notified about AWS Ground Station events. You receive a notification when an event matches a rule that you specify. You can receive notifications for events through multiple channels, including email, AWS Chatbot chat notifications, or AWS Console Mobile Application push notifications. You can also see notifications in the Console Notifications Center. User Notifications supports aggregation, which can reduce the number of notifications you receive during specific events.

Use the following topics to monitor AWS Ground Station.

Topics

- Logging AWS Ground Station API Calls with AWS CloudTrail
- Metrics with Amazon CloudWatch

Automating AWS Ground Station with Events

Note

This document uses the term “event” throughout. CloudWatch Events and EventBridge are the same underlying service and API. Rules to match incoming events and route them to targets for processing can be built using either service.

Events enable 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 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 events with AWS Ground Station include:

- Invoking a Lambda function to automate the starting and stopping of Amazon EC2 instances based off the event state.
- Publishing to an Amazon SNS topic whenever a contact changes states. These topics can be set up to send out email notices at the beginning or end of contacts.

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

Example Events

> **Note**
>
> All events generated by AWS Ground Station have "aws.groundstation" as the value for "source".

Ground Station Contact State Change

If you want to perform a specific action when an upcoming contact is changing states, you can setup a rule to automate this action. This is helpful for when you want to receive notifications about the state changes of your contact. If you would like to change when you receive these events, you can modify your mission profile's contactPrePassDurationSeconds and contactPostPassDurationSeconds. The events are sent to the region that the contact was scheduled from.

An example is provided below.

```
{
    "version": "0",
    "id": "01234567-0123-0123",
    "account": "123456789012",
    "time": "2019-05-30T17:40:30Z",
    "region": "us-west-2",
    "source": "aws.groundstation",
    "resources": [
        "arn:aws:groundstation:us-
west-2:123456789012:contact/11111111-1111-1111-1111-111111111111"
    ],
    "detailType": "Ground Station Contact State Change",
    "detail": {
        "contactId": "11111111-1111-1111-1111-111111111111",
        "groundstationId": "Ground Station 1",
        "missionProfileArn": "arn:aws:groundstation:us-west-2:123456789012:mission-
profile/11111111-1111-1111-1111-111111111111",
        "satelliteArn":
 "arn:aws:groundstation::123456789012:satellite/11111111-1111-1111-1111-111111111111",
        "contactStatus": "PASS"
    },
    "account": "123456789012"
}
```

The possible values for `contactStatus` are defined in the section called “Ground Station Contact Statuses”.

Ground Station Dataflow Endpoint Group State Change

If you want to perform an action when your dataflow endpoint group is being used to receive data, you can set up a rule to automate this action. This will allow you to perform different actions in response to the dataflow endpoint group status changing states. If you would like to change when you receive these events, use a dataflow endpoint group with different contactPrePassDurationSeconds and contactPostPassDurationSeconds. This event will be sent to the region of the dataflow endpoint group.

An example is provided below.

```
{
    "version": "0",
    "id": "01234567-0123-0123",
    "account": "123456789012",
    "time": "2019-05-30T17:40:30Z",
    "region": "us-west-2",
    "source": "aws.groundstation",
    "resources": [
        "arn:aws:groundstation:us-west-2:123456789012:dataflow-endpoint-
group/bad957a8-1d60-4c45-a92a-39febd98921d, arn:aws:groundstation:us-
west-2:123456789012:contact/98ddd10f-f2bc-479c-bf7d-55644737fb09,
 arn:aws:groundstation:us-west-2:123456789012:mission-profile/c513c84c-eb40-4473-88a2-
d482648c9234"
    ],
    "detailType": "Ground Station Dataflow Endpoint Group State Change",
    "detail": {
        "dataflowEndpointGroupId": "bad957a8-1d60-4c45-a92a-39febd98921d",
        "groundstationId": "Ground Station 1",
        "contactId": "98ddd10f-f2bc-479c-bf7d-55644737fb09",
        "dataflowEndpointGroupArn": "arn:aws:groundstation:us-
west-2:680367718957:dataflow-endpoint-group/bad957a8-1d60-4c45-a92a-39febd98921d",
        "missionProfileArn": "arn:aws:groundstation:us-west-2:123456789012:mission-
profile/c513c84c-eb40-4473-88a2-d482648c9234",
        "dataflowEndpointGroupState": "PREPASS"
    },
    "account": "123456789012"
}
```

Possible states for the `dataflowEndpointGroupState` include `PREPASS`, `PASS`, `POSTPASS`, and `COMPLETED`.

Ground Station Ephemeris State Change

If you want to perform an action when an ephemeris changes state, you can set up a rule to automate this action. This allows you to perform different actions in response to an ephemeris changing state. For example, you can perform an action when an ephemeris has completed validation, and it is now `ENABLED`. Notification for this event will be sent to the region were the ephemeris was uploaded.

An example is provided below.

```
    {
    "id": "7bf73129-1428-4cd3-a780-95db273d1602",
    "detail-type": "Ground Station Ephemeris State Change",
    "source": "aws.groundstation",
    "account": "123456789012",
    "time": "2019-12-03T21:29:54Z",
    "region": "us-west-2",
    "resources": [
    "arn:aws:groundstation::123456789012:satellite/10313191-c9d9-4ecb-a5f2-
bc55cab050ec",
    "arn:aws:groundstation::123456789012:ephemeris/111111-cccc-bbbb-a555-bcccca005000",
    ],
    "detail": {
    "ephemerisStatus": "ENABLED",
    "ephemerisId": "111111-cccc-bbbb-a555-bcccca005000",
    "satelliteId": "10313191-c9d9-4ecb-a5f2-bc55cab050ec"
    }
    }
```

Possible states for the `ephemerisStatus` include `ENABLED`, `VALIDATING`, `INVALID`, `ERROR`, `DISABLED`, `EXPIRED`

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 `ReserveContact` 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"
}
```

Metrics with Amazon CloudWatch

During a contact, AWS Ground Station automatically captures and sends data to CloudWatch for analysis. Your data can be viewed on a graph or as source code in the Amazon CloudWatch console. For more information about accessing and CloudWatch Metrics, see Using Amazon CloudWatch Metrics.

AWS Ground Station Metrics and Dimensions

What metrics are available?

The following metrics are available from AWS Ground Station.

Metric	Description
`AzimuthAngle`	The azimuth angle of the antenna. True north is 0 degrees and east is 90 degrees. Units: degrees
`BitErrorRate`	The error rate on bits in a given number of bit transmissions. Bit errors are caused by noise, distortion, or interference Units: Bits errors per unit time
`BlockErrorRate`	The error rate of blocks in a given number of received blocks. Block errors are caused by interference. Units: Erroneous blocks / Total number of blocks
`CarrierFrequencyRecovery_Cn0`	Carrier to noise density ratio per unit bandwidth. Units: decibel-Hertz (dB-Hz)
`CarrierFrequencyRecovery_Locked`	Set to 1 when the demodulator carrier frequency recovery loop is locked and 0 when unlocked. Units: unitless
`CarrierFrequencyRecovery_OffsetFrequency_Hz`	The offset between the estimated signal center and ideal center frequency. This is caused by Doppler shift and local oscillator offset between spacecraft and antenna system. Units: hertz (Hz)
`ElevationAngle`	The elevation angle of the antenna. The horizon is 0 degrees and zenith is 90 degrees. Units: degrees

Metric	Description
`Es/N0`	The ratio of energy per symbol to noise power spectral density. Units: decibels (dB)
`ReceivedPower`	The measured signal strength in the demodulator/ decoder. Units: decibels relative to milliwatts (dBm)
`SymbolTimingRecovery_ErrorVectorMagnitude`	The error vector magnitude between received symbols and ideal constellation points. Units: percent
`SymbolTimingRecovery_Locked`	Set to 1 when the demodulator symbol timing recovery loop is locked and 0 when unlocked Units: unitless
`SymbolTimingRecovery_OffsetSymbolRate`	The offset between the estimated symbol rate and ideal signal symbol rate. This is caused by Doppler shift and local oscillator offset between spacecraft and antenna system. Units: symbols/second

What dimensions are used for AWS Ground Station?

You can filter AWS Ground Station data using the following dimensions.

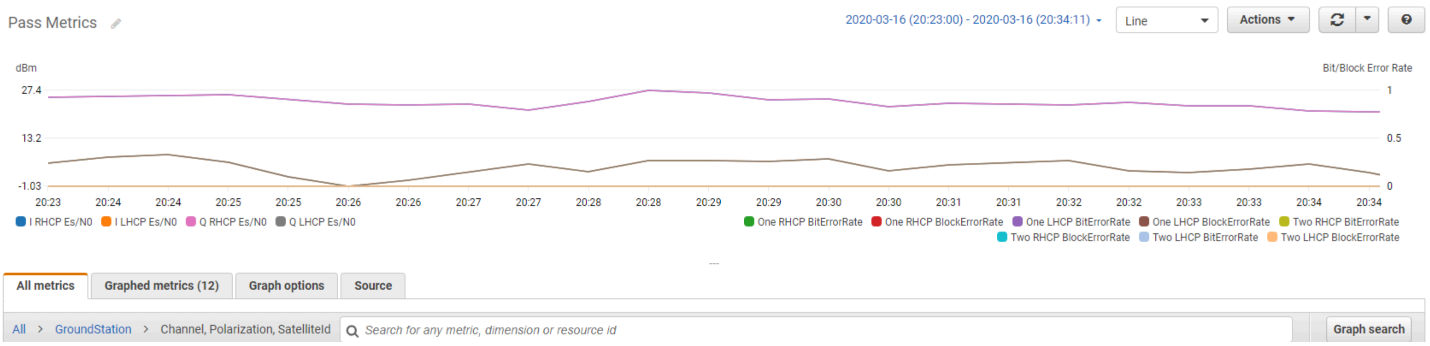
Dimension	Description
`Channel`	The channels for each contact include One, Two, I (in-phase), and Q (quadrature).

Dimension	Description
`Polarization`	The polarization for each contact include LHCP (Left Hand Circular Polarized) or RHCP (Right Hand Circular Polarized).
`SatelliteId`	The satellite ID contains the ARN of the satellite for your contacts.

Viewing Metrics

When viewing graphed metrics, it is important to note that the aggregation window determines how your metrics will be displayed. Each metric in a contact can be displayed as data per second for 3 hours after the data is received. Your data will be aggregated by CloudWatch Metrics as data per minute after that 3 hour period has elapsed. If you need to view your metrics on a data per second measurement, it is recommended to view your data within the 3 hour period after the data is received or persist it outside of CloudWatch Metrics.

In addition, any data captured within the first 60 seconds will not contain enough information to produce meaningful metrics, and will likely not be displayed. In order to view meaningful metrics, it is recommended to view your data after 60 seconds has passed.

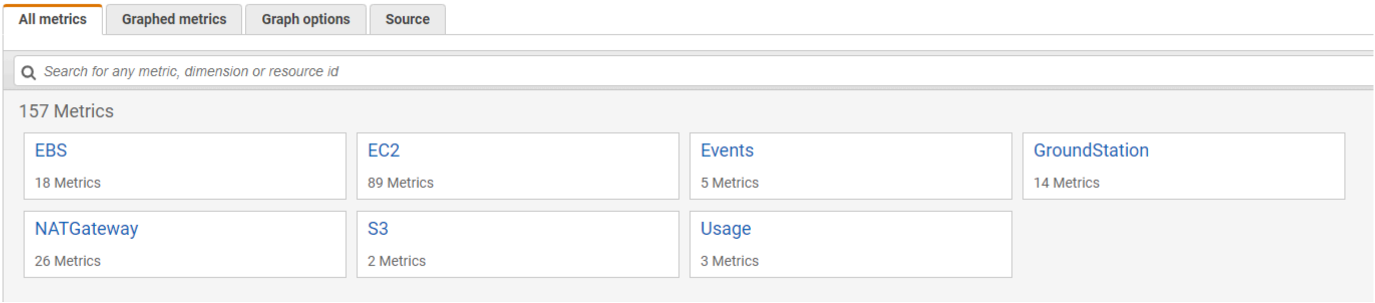


For more information about graphing AWS Ground Station metrics in CloudWatch, see Graphing Metrics.

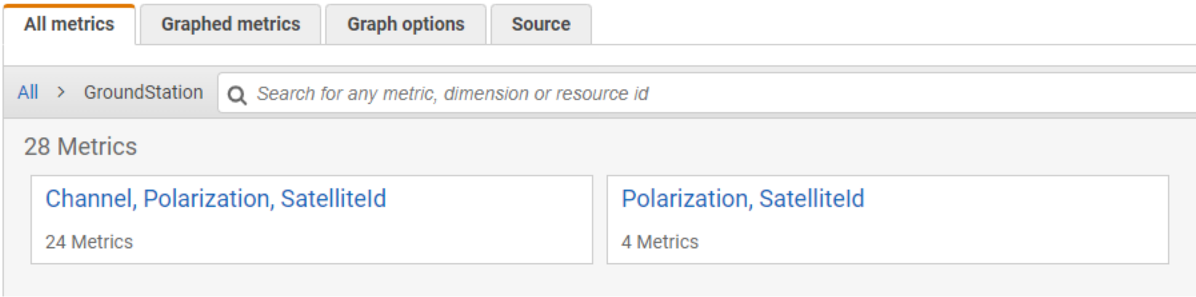
To view metrics using the console

1. Open the CloudWatch console.
2. In the navigation pane, choose **Metrics**.

3. Select the **GroundStation** namespace.



4. Select your desired metric dimensions (for example, **Channel, Polarization, SatelliteId**.


All metrics
Graphed metrics
Graph options
Source
All > GroundStation
Search for any metric, dimension or resource id
28 Metrics
Channel, Polarization, SatelliteId
24 Metrics
Polarization, SatelliteId
4 Metrics

5. The **All metrics** tab displays all metrics for that dimension in the namespace. You can do the following:

   a. To sort the table, use the column heading.

   b. To graph a metric, select the check box associated with the metric. To select all metrics, select the check box in the heading row of the table.

   c. To filter by resource, choose the resource ID and then choose **Add to search**.

   d. To filter by metric, choose the metric name and then choose **Add to search**.

To view metrics using AWS CLI

1. Ensure that AWS CLI is installed. For information about installing AWS CLI, see Installing the AWS CLI.

2. Create a CloudWatch agent configuration JSON file. For instructions on creating a CloudWatch agent configuration file, see Create the CloudWatch Agent Configuration File.

3. List the available CloudWatch metrics by running `aws cloudwatch list-metrics`.

4. Modify the JSON file you created in step 2 to match the `SatellitID` from your metrics.

> **Note**
>
> Do not reduce the `Period` field to a value under 60. AWS Ground Station publishes metrics every 60 seconds and no metrics will be returned if the value is reduced.

5. Run `aws cloudwatch get-metric-data` with time periods of your passes and your CloudWatch agent configuration JSON file. An example is provided below.

```
aws cloudwatch get-metrics-data -start-time 2020-02-26T19:12:00Z -end-time
 2020-02-26T19:24:00Z -metric-data-queries file://metricdata.json
```

Metrics will be provided with timestamps from your contact. An example output of AWS Ground Station metrics is provided below.

```
{
      “MetricDataResults”: [
          {
              “Id”: “myQuery”,
              “Label”: “Es/N0”,
              “Timestamps”: [
                  “2020-02-18T19:44:00Z”,
                  “2020-02-18T19:43:00Z”,
                  “2020-02-18T19:42:00Z”,
                  “2020-02-18T19:41:00Z”,
                  “2020-02-18T19:40:00Z”,
                  “2020-02-18T19:39:00Z”,
                  “2020-02-18T19:38:00Z”,
                  “2020-02-18T19:37:00Z”,
              ],
              “Values”: [
                  24.58344556958329,
                  24.251638725562216,
                  22.919391450230158,
                  22.83838908204037,
                  23.303086848486842,
                  22.845261784583364,
                  21.34531397048953,
                  19.171561698261222
              ],
              “StatusCode”: “Complete”
          }
      ]
      “Messages”: []
}
```

Troubleshooting

The following documentation can help you troubleshoot issues that may prevent an AWS Ground Station contact from completing successfully.

Topics

Troubleshooting Contacts that Deliver Data to Amazon EC2

If you are unable to successfully complete an AWS Ground Station contact, you will need to verify that your Amazon EC2 instance is running, verify that Data Defender is running, and verify that your Data Defender stream is configured properly.

Prerequisite

The following procedures assume that an Amazon EC2 instance is already set up. To set up an Amazon EC2 instance in AWS Ground Station, see Getting Started.

Step 1: Verify that Your EC2 Instance is Running

1. Locate the Amazon EC2 instance that was used for the contact you are troubleshooting. Use the following steps:

    a. In your **CloudFormation** dashboard, select the stack that contains your Amazon EC2 instance.

    b. Choose the **Resources** tab and locate your Amazon EC2 instance in the **Logical ID** column. Verify that the instance is created in the **Status** column.

    c. In the **Physical ID** column, choose the link for your Amazon EC2 instance. This will take you to the Amazon EC2 management console.

2. In the Amazon EC2 management console, ensure that your Amazon EC2 **Instance State** is *running*.

3. If your instance is running, continue to the next step. If your instance is not running, start the instance by using the following step:

    - With your Amazon EC2 instance selected, choose **Actions > Instance State > Start**.

Step 2: Determine Type of Dataflow Application Used

If you are using the **AWS Ground Station Agent** for data delivery please redirect to section Troubleshooting AWS Ground Station Agent.

Otherwise if you are using the **Data Defender (DDX)** application continue to the section called "Step 3: Verify that Data Defender is Running".

Step 3: Verify that Data Defender is Running

Verifying the status of Data Defender requires you to connect to your instance in Amazon EC2. For more details on connecting to your instance, see Connect to Your Linux Instance.

The following procedure provides troubleshooting steps using commands in an SSH client.

1. Open a terminal or command prompt and connect to your Amazon EC2 instance by using SSH. Forward port 80 of the remote host in order to view the Data Defender web UI. The following commands demonstrate how to use SSH to connect to an Amazon EC2 instance through a bastion with port forwarding enabled.

    > **Note**
    >
    > You must replace <SSH KEY>, <BASTION HOST>, and <HOST> with your specific ssh key, bastion host name, and Amazon EC2 instance host name.

    For Windows

    ```
    ssh -L 8080:localhost:80 -o ProxyCommand="C:\Windows\System32\OpenSSH\ssh.exe -o
     \"ForwardAgent yes\" -W %h:%p -i \"<SSH KEY>\" ec2-user@<BASTION HOST>" -i "<SSH
     KEY>" ec2-user@<HOST>
    ```

    For Mac

```
ssh -L 8080:localhost:80 -o ProxyCommand="ssh -A -o 'ForwardAgent yes' -W %h:%p -i
 <SSH KEY> ec2-user@<BASTION HOST>" -i <SSH  KEY> ec2-user@<HOST>
```

2. Verify that Data Defender (also called DDX) is running by grepping (checking) for a running process named ddx in the output. The command for grepping (checking) for a running process and a successful example output is provided below.

   ```
   [ec2-user@Receiver-Instance ~]$ ps -ef | grep ddx
   Rtlogic   4977      1 10 Oct16 ?        2-00:22:14 /opt/rtlogic/ddx/bin/ddx -m/
   opt/rtlogic/ddx/modules -p/opt/rtlogic/ddx/plugins -c/opt/rtlogic/ddx/bin/ddx.xml -
   umask=077 -daemon -f installed=true -f security=true -f enable HttpsForwarding=true
   Ec2-user 18787 18657  0 16:51 pts/0      00:00:00 grep -color=auto ddx
   ```

   If Data Defender is running, skip to the section called “Step 4: Verify that Your Data Defender Stream is Configured” Otherwise, continue to the next step.

3. Start Data Defender using the command show below.

   ```
   sudo service rtlogic-ddx start
   ```

   If Data Defender is running after using the command, skip to the section called “Step 4: Verify that Your Data Defender Stream is Configured” Otherwise, continue to the next step.

4. Inspect the following files using the commands below to see if there were any errors while installing and configuring Data Defender.

   ```
   cat /var/log/user-data.log
   cat /opt/aws/groundstation/.startup.out
   ```

   > **Note**
   >
   > A common issue discovered when inspecting these files is that the Amazon VPC that your Amazon EC2 instance is running in does not have access to Amazon S3 to download the installation files. If you discover in your logs that this is the issue, check your EC2 instance's Amazon VPC and security group settings to ensure they are not blocking access to Amazon S3.

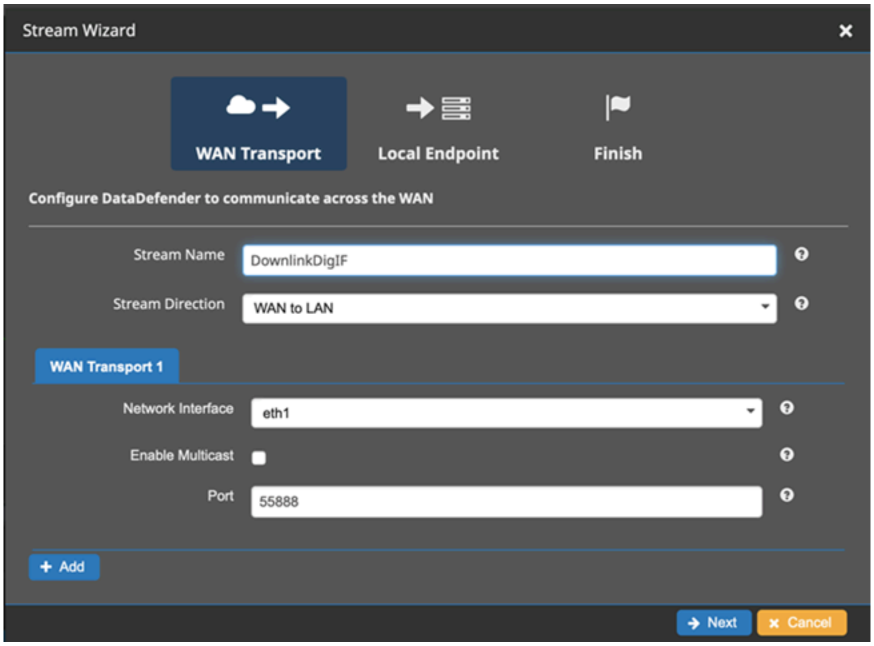
If Data Defender is running after checking your Amazon VPC settings, continue to the section called "Step 4: Verify that Your Data Defender Stream is Configured". If the problem persists, contact AWS Support and send your log files with a description of your issue.

Step 4: Verify that Your Data Defender Stream is Configured

1. In a web browser, access your DDX Web User Interface by entering the following address in the address bar: *localhost:8080*. Then, press **Enter**.
2. On the **DataDefender** dashboard, choose **Go to Details**.
3. Select your stream from the list of streams, and choose **Edit Stream**.
4. In the **Stream Wizard** dialog box, do the following:

    a. In the **WAN Transport** pane, ensure **WAN to LAN** is selected for **Stream Direction**.

    b. In the **Port** box, ensure the WAN port you have chosen for your dataflow endpoint group is present. By default, this port is 55888. Then, choose **Next**.



Stream Wizard

Configure DataDefender to communicate across the WAN

Stream Name: DownlinkDigIF

Stream Direction: WAN to LAN

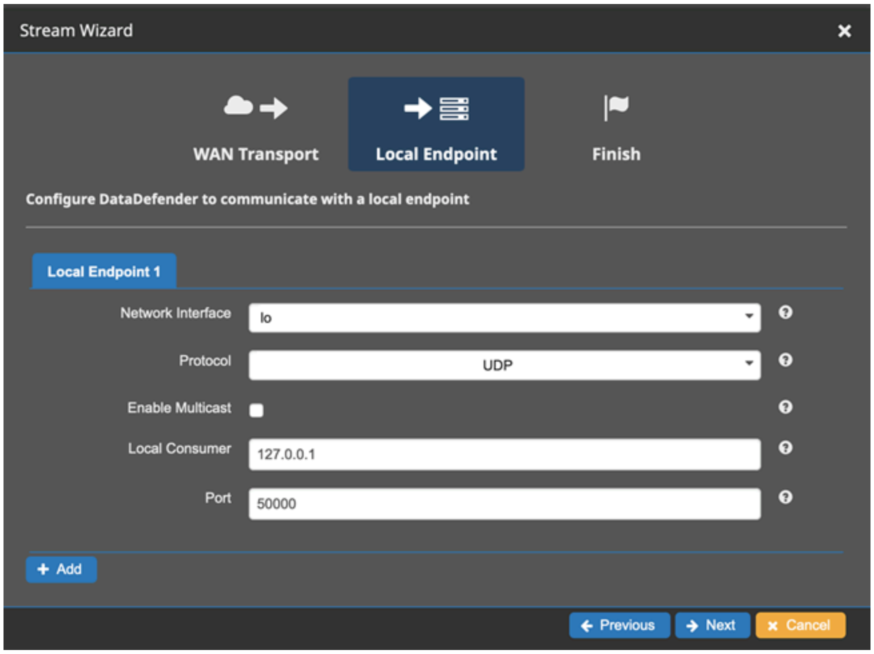
WAN Transport 1

Network Interface: eth1

Enable Multicast □

Port: 55888

    c. In the **Local Endpoint** pane, ensure that a valid port is present in the *Port* box. By default, this port is 50000. This is the port on which you will receive your data after Data Defender has received it from the AWS Ground Station service. Then, choose **Next**.



Configure DataDefender to communicate with a local endpoint

Network Interface: lo

Protocol: UDP

Enable Multicast: □

Local Consumer: 127.0.0.1

Port: 50000

d. Choose **Finish** in the remaining menu if you have changed any values. Otherwise, you can cancel out of the **Stream Wizard** menu.

You have now ensured that your Amazon EC2 instance and Data Defender are both running and configured properly to receive data from AWS Ground Station. If you continue experience issues, contact AWS Support.

Ground Station Contact Statuses

The status of an AWS Ground Station contact provides insight into what is happening to that contact at a given time.

Contact Statuses

The following is the list of statuses that a contact can have:

- **AVAILABLE** - The contact is available to be reserved.
- **SCHEDULING** - The contact is in the process of scheduling.
- **SCHEDULED** - The contact was successfully scheduled.
- **FAILED_TO_SCHEDULE** - The contact failed to schedule.
- **PREPASS** - The contact is starting soon and resources are being prepared.

- **PASS** - The contact is currently executing and the satellite is being communicated with.
- **POSTPASS** - The communication has completed and resources used are being cleaned up.
- **COMPLETED** - The contact completed successfully.
- **FAILED** - The contact failed because of an issue with the customer resource configuration.
- **AWS_FAILED** - The contact failed because of a problem in the AWS Ground Station service.
- **CANCELLING** - The contact is in the process of being cancelled.
- **AWS_CANCELLED** - The contact was cancelled by the AWS Ground Station service. Antenna or site maintenance is one example of when this could happen.
- **CANCELLED** - The contact was cancelled by the customer.

Troubleshooting Guides

- the section called "Troubleshooting FAILED Contacts"
- the section called "Troubleshooting FAILED_TO_SCHEDULE Contacts"

Troubleshooting FAILED Contacts

A contact will have a terminal contact status of **FAILED** when AWS Ground Station detected an issue with the customer resource configuration. The common use cases that can cause **FAILED** contacts are provided below, along with steps to help troubleshoot.

> **Note**
>
> This guide is specifically for the **FAILED** contact status - and is not intended for other failure statuses, such as **AWS_FAILED**, **AWS_CANCELLED**, or **FAILED_TO_SCHEDULE**. For more information on contact statuses, see the section called "Ground Station Contact Statuses"

Data Defender (DDX) FAILED Use Cases

The following is the list of common use cases that can result in a **FAILED** contact status for DDX-based dataflows:

- **Customer DDX Never Connects** - The DDX connection between AWS Ground Station Antenna and Customer Dataflow Endpoint Group for one or more dataflows was never established.
- **Customer DDX Connects Late** - The DDX connection between AWS Ground Station Antenna and Customer Dataflow Endpoint Group for one or more dataflows was established after the contact start time.

For any DDX dataflow failure cases, it is recommended to look into the following:

- Confirm the receiver Amazon EC2 instance was started successfully, prior to contact start time.
- Confirm the DDX was up and running during the contact.

See the section on the section called "Troubleshooting Contacts that Deliver Data to Amazon EC2" for more specific troubleshooting steps.

AWS Ground Station Agent FAILED Use Cases

The following is the list of common use cases that can result in a **FAILED** contact status for Agent-based dataflows:

- **Customer Agent Never Reported Status** - The Agent responsible for orchestrating data delivery on the Customer Dataflow Endpoint Group for one or more dataflows never successfully reported status to AWS Ground Station. This status update should happen within a few seconds of the contact end time.
- **Customer Agent Started Late** - The Agent responsible for orchestrating data delivery on the Customer Dataflow Endpoint Group for one or more dataflows was started late, after the contact start time.

For any AWS Ground Station Agent dataflow failure cases, it is recommended to look into the following:

- Confirm the receiver Amazon EC2 instance was started successfully, prior to contact start time.
- Confirm the Agent application was up and running at the start and during the contact.
- Confirm the Agent application and Amazon EC2 instance were not shut down within 15 seconds of contact end. This provides the Agent sufficient time to report status to AWS Ground Station.

See the section on the section called “Troubleshooting Contacts that Deliver Data to Amazon EC2” for more specific troubleshooting steps.

Troubleshooting FAILED_TO_SCHEDULE Contacts

A contact will **FAILED_TO_SCHEDULE** when AWS Ground Station detected an issue either with the customer resource configuration or within the internal system. A contact that ends in a **FAILED_TO_SCHEDULE** state will optionally provide an `errorMessage` for additional context. For information about describing contacts, see the section called “Describe a Contact with AWS CLI”.

The common use cases that can cause **FAILED_TO_SCHEDULE** contacts are provided below, along with steps to help troubleshoot.

> **Note**
>
> This guide is specifically for the **FAILED_TO_SCHEDULE** contact status - and is not intended for other failure statuses, such as **AWS_FAILED**, **AWS_CANCELLED**, or **FAILED**. For more information on contact statuses, see the section called “Ground Station Contact Statuses”

The settings specified in your Antenna Downlink Demod Decode Config are not supported

The mission profile that was used to schedule this contact had an antenna-downlink-demod-decode config that was not valid.

Previously existing AntennaDownlinkDemodDecode config

- If your antenna-downlink-demod-decode configs have recently been changed - roll back to a previously working version before attempting to schedule.
- If this was an intentional change on an existing config, or a previously existing config that is no longer successfully scheduling - follow the next step on how to onboard a new AntennaDownlinkDemodDecode config.

Newly created AntennaDownlinkDemodDecode config

Contact AWS Ground Station directly to onboard your new config. Create a case with AWS Support including the `contactId` that ended in the **FAILED_TO_SCHEDULE** state

General Troubleshooting Steps

If the preceding troubleshooting steps did not resolve your issue:

- Re-attempt scheduling the contact or schedule another contact using the same mission profile. See the section called “Reserve a Contact with AWS CLI”.
- If you continue to receive a **FAILED_TO_SCHEDULE** status for this mission profile, contact AWS Support

Security in AWS Ground Station

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. AWS provides security-specific tools and features to help you meet your security objectives. These tools and features include network security, configuration management, access control, and data security.

When using AWS Ground Station, we recommend that you follow industry best practices and implement end-to-end encryption. AWS provides APIs for you to integrate encryption and data protection. For more information about AWS security, see the Introduction to AWS Security whitepaper.

Use the following topics to learn how to secure your resources.

Topics

- Identity and Access Management for AWS Ground Station
- Using service-linked roles for Ground Station
- AWS managed policies for AWS Ground Station

Identity and Access Management 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 an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS Ground Station works with IAM
- Identity-based policy examples for AWS Ground Station
- Troubleshooting AWS Ground Station identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that 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 how access is managed 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 identity and access.

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 service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS Ground Station, see How AWS Ground Station works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS Ground Station. To view example AWS Ground Station identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Ground Station.

Authenticating with identities

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

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

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

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

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

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

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

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

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

IAM users and groups

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

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

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see 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 AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances 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 AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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

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

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

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see 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 an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If

you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.

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

Multiple policy types

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

How AWS Ground Station works with IAM

Before you use IAM to manage access to AWS Ground Station, learn what IAM features are available to use with AWS Ground Station.

IAM features you can use with AWS Ground Station

IAM feature	AWS Ground Station support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes

IAM feature	AWS Ground Station support
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

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

Identity-based policies for AWS Ground Station

Supports identity-based policies	Yes

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

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

Identity-based policy examples for AWS Ground Station

To view examples of AWS Ground Station identity-based policies, see Identity-based policy examples for AWS Ground Station.

Resource-based policies within AWS Ground Station

Supports resource-based policies	No

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

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for AWS Ground Station

Supports policy actions	Yes

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

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

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

To see a list of AWS Ground Station actions, see Actions defined by AWS Ground Station in the *Service Authorization Reference*.

Policy actions in AWS Ground Station use the following prefix before the action:

```
groundstation
```

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

```
"Action": [
      "groundstation:action1",
      "groundstation:action2"
         ]
```

To view examples of AWS Ground Station identity-based policies, see Identity-based policy examples for AWS Ground Station.

Policy resources for AWS Ground Station

Supports policy resources	Yes

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

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

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

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

To see a list of AWS Ground Station resource types and their ARNs, see Resources defined by AWS Ground Station in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Ground Station.

To view examples of AWS Ground Station identity-based policies, see Identity-based policy examples for AWS Ground Station.

Policy condition keys for AWS Ground Station

Supports service-specific policy condition keys	Yes

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

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

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

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

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

To see a list of AWS Ground Station condition keys, see Condition keys for AWS Ground Station in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Ground Station.

To view examples of AWS Ground Station identity-based policies, see Identity-based policy examples for AWS Ground Station.

ACLs in AWS Ground Station

Supports ACLs	No

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

ABAC with AWS Ground Station

Supports ABAC (tags in policies)	Yes

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

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

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

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

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

Using temporary credentials with AWS Ground Station

Supports temporary credentials	Yes

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

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

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

Cross-service principal permissions for AWS Ground Station

Supports forward access sessions (FAS)	Yes

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

Service roles for AWS Ground Station

Supports service roles	No

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

> ⚠ **Warning**
>
> Changing the permissions for a service role might break AWS Ground Station functionality. Edit service roles only when AWS Ground Station provides guidance to do so.

Service-linked roles for AWS Ground Station

Supports service-linked roles	Yes

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

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

Identity-based policy examples for AWS Ground Station

By default, users and roles don't have permission to create or modify AWS Ground Station resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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

For details about actions and resource types defined by AWS Ground Station, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Ground Station in the *Service Authorization Reference*.

Topics

Policy best practices

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

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM 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 policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

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

To ensure that users and roles can still use the AWS Ground Station console, also attach the AWS Ground Station *`ConsoleAccess`* or *`ReadOnly`* AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

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

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

Troubleshooting AWS Ground Station identity and access

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

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

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

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `groundstation:`*`GetWidget`* permissions.

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

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `groundstation:`*`GetWidget`* action.

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

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to 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 that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

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

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

I want to allow people outside of my AWS account to access my 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.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.

- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Using service-linked roles for Ground Station

AWS Ground Station uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Ground Station. Service-linked roles are predefined by Ground Station and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Ground Station easier because you don't have to manually add the necessary permissions. Ground Station defines the permissions of its service-linked roles, and unless defined otherwise, only Ground Station can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Ground Station

Ground Station uses the service-linked role named **AWSServiceRoleForGroundStationDataflowEndpointGroup** – AWS GroundStation uses this service-linked role to invoke EC2 to find public IPv4 addresses.

The AWSServiceRoleForGroundStationDataflowEndpointGroup service-linked role trusts the following services to assume the role:

- `groundstation.amazonaws.com`

The role permissions policy named AWSServiceRoleForGroundStationDataflowEndpointGroupPolicy allows Ground Station to complete the following actions on the specified resources:

- Action: `ec2:DescribeAddresses` on `all AWS resources (*)`

  Action allows Ground Station to list all IPs associated with EIPs.
- Action: `ec2:DescribeNetworkInterfaces` on `all AWS resources (*)`

  Action allows Ground Station to get information on the network interfaces associated with EC2 instances

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for Ground Station

You don't need to manually create a service-linked role. When you create a DataflowEndpointGroup in the AWS CLI or the AWS API, Ground Station creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a DataflowEndpointGroup, Ground Station creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **Data Delivery to Amazon EC2** use case. In the AWS CLI or the AWS API, create a service-linked role with the `groundstation.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for Ground Station

Ground Station does not allow you to edit the AWSServiceRoleForGroundStationDataflowEndpointGroup service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for Ground Station

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained.

You can delete a service-linked role only after first deleting the DataflowEndpointGroups using the service-linked role. This protects you from inadvertently revoking permissions to your DataflowEndpointGroups. If a service-linked role is used with multiple DataflowEndpointGroups, you must delete all DataflowEndpointGroups that use the service-linked role before you can delete it.

> **Note**
>
> If the Ground Station service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete Ground Station resources used by the AWSServiceRoleForGroundStationDataflowEndpointGroup

- Delete DataflowEndpointGroups via the AWS CLI or the AWS API.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForGroundStationDataflowEndpointGroup service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported regions for Ground Station service-linked roles

Ground Station supports using service-linked roles in all of the regions where the service is available. For more information, see Region Table.

Troubleshooting

`NOT_AUTHORIZED_TO_CREATE_SLR` - This indicates the role in your account that is being used to call the CreateDataflowEndpointGroup API does not have the `iam:CreateServiceLinkedRole` permission. An administrator with the `iam:CreateServiceLinkedRole` permission must manually create the Service-Linked Role for your account.

AWS managed policies for AWS Ground Station

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

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

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

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

AWS managed policy: AWSGroundStationAgentInstancePolicy

You can attach the `AWSGroundStationAgentInstancePolicy` policy to your IAM identities.

This policy grants AWS Ground Station Agent permissions to a customer instance that allows the instance to send and receive data during Ground Station contacts. All permissions in this policy are from the Ground Station service.

Permissions details

This policy includes the following permissions.

- `groundstation` – Allows dataflow endpoint instances to call the Ground Station Agent APIs.

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

AWS managed policy: AWSServiceRoleForGroundStationDataflowEndpointGroupPolicy

You can't attach AWSServiceRoleForGroundStationDataflowEndpointGroupPolicy to your IAM entities. This policy is attached to a service-linked role that allows AWS Ground Station to perform actions on your behalf. For more information, see Using service linked roles.

This policy grants EC2 permissions that allow AWS Ground Station to find public IPv4 addresses.

Permissions details

This policy includes the following permissions.

- `ec2:DescribeAddresses` – Allows AWS Ground Station to list all IPs associated with EIPs on your behalf.
- `ec2:DescribeNetworkInterfaces` – Allows AWS Ground Station to get information on the network interfaces associated with EC2 instances on your behalf.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAddresses",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Ground Station updates to AWS managed policies

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

Change	Description	Date
AWSGroundStationAgentInstancePolicy – New policy	AWS Ground Station added a new policy to provide the dataflow endpoint instance permissions to use the AWS Ground Station Agent.	April 12, 2023

Change	Description	Date
AWSServiceRoleForGroundStationDataflowEndpointGroupPolicy – New policy	AWS Ground Station added a new policy that grants EC2 permissions to allow AWS Ground Station to find public IPv4 addresses associated with EIPs and network interfaces associated with EC2 instances.	November 02, 2022
AWS Ground Station started tracking changes	AWS Ground Station started tracking changes for AWS managed policies.	March 01, 2021

Data Encryption at rest for AWS Ground Station

AWS Ground Station provides encryption by default to protect sensitive customer data at rest using AWS owned encryption keys.

- *AWS owned keys* - AWS Ground Station uses these keys by default to automatically encrypt personal, directly identifiable data and ephemerides. You cannot view, manage, or use AWS-owned keys, or audit their use; however, it is unnecessary to take any action or change programs to protect the keys that encrypt data. For more information, see AWS-owned keys in the AWS Key Management Service Developer Guide.

Encryption of data at rest by default helps by reducing the operational overhead and complexity involved in protecting sensitive data. At the same time, it enables building secure applications that meet strict encryption compliance, as well as regulatory requirements.

AWS Ground Station enforces encryption on all sensitive, at-rest, data, however, for some AWS Ground Station resource, such as ephemerides, you can choose to use a customer managed key in place of the default AWS managed keys.

- *Customer managed keys* -- AWS Ground Station supports the use of a symmetric customer managed key that you create, own, and manage to add a second layer of encryption over the existing AWS owned encryption. Because you have full control of this layer of encryption, you can perform such tasks as:
  - Establishing and maintaining key policies
  - Establishing and maintaining IAM policies and grants
  - Enabling and disabling key policies
  - Rotating key cryptographic material
  - Adding tags
  - Creating key aliases
  - Scheduling keys for deletion

  For more information, see customer managed key in the AWS Key Management Service Developer Guide.

The following table summarizes resources for which AWS Ground Station supports the use of Customer Managed Keys

Data type	AWS owned key encryption	Customer managed key encryption (Optional)
Ephemeris data used to compute the trajectory of a Satellite	Enabled	Enabled

> **Note**
>
> AWS Ground Station automatically enables encryption at rest using AWS owned keys to protect personally identifiable data at no charge. However, AWS KMS charges apply for using a customer managed key. For more information about pricing, see the AWS Key Management Service pricing.
> For more information on AWS KMS, see the AWS KMS Developer Guide.

How AWS Ground Station uses grants in AWS KMS

AWS Ground Station requires a key grant to use your customer-managed key.

When you upload an ephemeris encrypted with a customer managed key, AWS Ground Station creates a key grant on your behalf by sending a CreateGrant request to AWS KMS. Grants in AWS KMS are used to give AWS Ground Station access to a KMS key in a customer account.

AWS Ground Station requires the grant to use your customer managed key for the following internal operations:

- Send `GenerateDataKey` requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send `Decrypt` requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.
- Send `Encrypt` requests to AWS KMS to encrypt the provided data.

You can revoke access to the grant, or remove the service's access to the customer managed key at any time. If you do, AWS Ground Station won't be able to access any of the data encrypted by the customer managed key, which affects operations that are dependent on that data. For example, if you remove a key grant from an ephemeris currently in use for a contact then AWS Ground Station will be unable to use the provided ephemeris data for pointing the antenna during the contact. This will cause the contact to end in a FAILED state.

Create a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs.

To create a symmetric customer managed key

Follow the steps for Creating symmetric customer managed key in the AWS Key Management Service Developer Guide.

Key policy

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see Managing access to customer managed keys in the AWS Key Management Service Developer Guide.

To use your customer managed key with your AWS Ground Station resources, the following API operations must be permitted in the key policy:

`kms:CreateGrant` - Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to grant operations AWS Ground Station requires. For more information about Using Grants, see the AWS Key Management Service Developer Guide.

This allows Amazon AWS to do the following:

- Call `GenerateDataKey` to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.
- Call `Encrypt` to use the data key to encrypt data.
- Set up a retiring principal to allow the service to `RetireGrant`.

`kms:DescribeKey` - Provides the customer managed key details to allow AWS Ground Station to validate the key before attempting to create a grant on the provided key.

The following are IAM policy statement examples you can add for AWS Ground Station

```
"Statement" : [
  {"Sid" : "Allow access to principals authorized to use AWS Ground Station",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "*"
    },
    "Action" : [
      "kms:DescribeKey",
      "kms:CreateGrant"
    ],
    "Resource" : "*",
    "Condition" : {
    "StringEquals" : {
        "kms:ViaService" : "groundstation.amazonaws.com",
        "kms:CallerAccount" : "111122223333"
    }
  },
  {"Sid": "Allow access for key administrators",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action" : [
      "kms:*"
      ],
    "Resource": "arn:aws:kms:region:111122223333:key/key_ID"
  },
  {"Sid" : "Allow read-only access to key metadata to the account",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : [
      "kms:Describe*",
      "kms:Get*",
      "kms:List*",
      "kms:RevokeGrant"
    ],
    "Resource" : "*"
  }
]
```

For more information about [specifying permissions in a policy](), see the AWS Key Management Service Developer Guide.

For more information about [troubleshooting key access](), see the AWS Key Management Service Developer Guide.

Specifying a customer managed key for AWS Ground Station

You can specify a customer managed key to encrypt the following resources:

- Ephemeris

When you create a resource, you can specify the data key by providing a *kmsKeyArn*

- *kmsKeyArn* - A [key identifier]() for an AWS KMS customer managed key

AWS Ground Station encryption context

An [encryption context]() is an optional set of key-value pairs that contain additional contextual information about the data. AWS KMS uses the encryption context as additional authenticated data to support authenticated encryption. When you include an encryption context in a request to encrypt data, AWS KMS binds the encryption context to the encrypted data. To decrypt data, you include the same encryption context in the request.

AWS Ground Station encryption context

AWS Ground Station uses the different encryption context depending on the resource being encrypted and specifies a specific encryption context for each key grant created.

Ephemeris Encryption Context:

Key grant for encrypting ephemeris resources are bound to a specific satellite ARN

```
"encryptionContext": {
    "aws:groundstation:arn":
 "arn:aws:groundstation::111122223333:satellite/00a770b0-082d-45a4-80ed-SAMPLE"
}
```

Note

Key grants are re-used for the same key-satellite pair.

Using encryption context for monitoring

When you use a symmetric customer managed key to encrypt your ephemerides, you can also use the encryption context in audit records and logs to identify how the customer managed key is being used. The encryption context also appears in logs generated by AWS CloudTrail or Amazon CloudWatch Logs .

Using encryption context to control access to your customer managed key

You can use the encryption context in key policies and IAM policies as `conditions` to control access to your symmetric customer managed key. You can also use encryption context constraints in a grant.

AWS Ground Station uses an encryption context constraint in grants to control access to the customer managed key in your account or region. The grant constraint requires that the operations that the grant allows use the specified encryption context.

The following are example key policy statements to grant access to a customer managed key for a specific encryption context. The condition in this policy statement requires that the grants have an encryption context constraint that specifies the encryption context.

```
{"Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:DescribeKey",
     "Resource": "*"
},{"Sid": "Enable CreateGrant",
     "Effect": "Allow",
     "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:CreateGrant",
     "Resource": "*",
     "Condition": {
        "StringEquals": {
            "kms:EncryptionContext:aws:groundstation:arn":
 "arn:aws:groundstation::111122223333:satellite/00a770b0-082d-45a4-80ed-SAMPLE"
        }
     }
}
```

Monitoring your encryption keys for AWS Ground Station

When you use an AWS KMS customer managed key with your AWS Ground Station resources, you can use AWS CloudTrail or Amazon CloudWatch logs to track requests that AWS Ground Station sends to AWS KMS. The following examples are AWS CloudTrail events for `CreateGrant`, `GenerateDataKey`, `Decrypt`, `Encrypt` and `DescribeKey` to monitor KMS operations called by AWS Ground Station to access data encrypted by your customer managed key.

`CreateGrant` (Cloudtrail)

When you use an AWS KMS customer managed key to encrypt your ephemeris resources, AWS Ground Station sends a `CreateGrant` request on your behalf to access the KMS key in your AWS account. The grant that AWS Ground Station creates are specific to the resource associated with the AWS KMS customer managed key. In addition, AWS Ground Station uses the `RetireGrant` operation to remove a grant when you delete a resource.

The following example event records the `CreateGrant` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AAAAAAAAAAAAAAAAAAAA:SampleUser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AAAAAAAAAAAAAAAAAAAA",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-02-22T22:22:22Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2022-02-22T22:22:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "111.11.11.11",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "operations": [
            "GenerateDataKeyWithoutPlaintext",
            "Decrypt",
            "Encrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:groundstation:arn":
 "arn:aws:groundstation::111122223333:satellite/00a770b0-082d-45a4-80ed-SAMPLE"
            }
        },
        "granteePrincipal": "groundstation.us-west-2.amazonaws.com",
        "retiringPrincipal": "groundstation.us-west-2.amazonaws.com",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DescribeKey (Cloudtrail)

When you use an AWS KMS customer managed key to encrypt your ephemeris resources, AWS Ground Station sends a DescribeKey request on your behalf to validate that the requested key exists in your account.

The following example event records the DescribeKey operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AAAAAAAAAAAAAAAAAAAAA:SampleUser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/User/Role",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AAAAAAAAAAAAAAAAAAAAA",
                "arn": "arn:aws:iam::111122223333:role/Role",
                "accountId": "111122223333",
                "userName": "User"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-02-22T22:22:22Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2022-02-22T22:22:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey (Cloudtrail)

When you use an AWS KMS customer managed key to encrypt your ephemeris resources, AWS Ground Station sends a `GenerateDataKey` request to KMS in order to generate a data key with which to encrypt your data.

The following example event records the `GenerateDataKey` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2022-02-22T22:22:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keySpec": "AES_256",
        "encryptionContext": {
            "aws:groundstation:arn":
 "arn:aws:groundstation::111122223333:satellite/00a770b0-082d-45a4-80ed-SAMPLE",
            "aws:s3:arn":
 "arn:aws:s3:::customerephemerisbucket/0034abcd-12ab-34cd-56ef-123456SAMPLE"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```

`Decrypt` (Cloudtrail)

When you use an AWS KMS customer managed key to encrypt your ephemeris resources, AWS Ground Station uses the `Decrypt` operation to decrypt the ephemeris provided if it is already encrypted with the same customer managed key. For example if an ephemeris is being uploaded from an S3 bucket and is encrypted in that bucket with a given key.

The following example event records the `Decrypt` operation:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2022-02-22T22:22:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionContext": {
            "aws:groundstation:arn":
 "arn:aws:groundstation::111122223333:satellite/00a770b0-082d-45a4-80ed-SAMPLE",
            "aws:s3:arn":
 "arn:aws:s3:::customerephemerisbucket/0034abcd-12ab-34cd-56ef-123456SAMPLE"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```

Satellite Ephemeris Data

An ephemeris, plural ephemerides, is a file or data structure providing the trajectory of astronomical objects. Historically, this file only referred to tabular data but, gradually, it has come to direct to a wide variety of data files indicating a spacecraft trajectory.

AWS Ground Station uses ephemeris data to determine when contacts become available for your satellite and correctly command antennas in the AWS Ground Station Network to point at your satellite. By default no action is required to provide AWS Ground Station with ephemerides.

Topics

Default Ephemeris Data

By default, AWS Ground Station uses publicly available data from Space-Track, and no action is required to supply AWS Ground Station with these default ephemerides. These ephemerides are two-line element sets associated with your satellite's NORAD ID. All default ephemerides have a priority of 0. As a result, they will be overridden, always, by any non-expired, custom ephemerides uploaded via the ephemeris API, which must always have a priority of 1, or greater.

Satellites without a NORAD ID, must upload custom ephemeris data to AWS Ground Station. For example, satellites that have just launched or that are intentionally omitted from the Space-Track catalog would have no NORAD ID and would need to have custom ephemerides uploaded. For more information on providing a custom ephemeris see: Providing Custom Ephemeris Data.

Which Ephemeris Is Used

Ephemerides have a *priority*, *expiration time*, and *enabled* flag. Together, these determine which ephemeris is used for a satellite. Only one ephemeris can be active for each satellite.

The ephemeris that will be used is the **highest-priority enabled ephemeris** whose expiration time is in the future. The available contact times returned by **ListContacts** are based on this ephemeris. If multiple `ENABLED` ephemerides have the same priority, the most recently created or updated ephemeris will be used.

> **Note**
>
> AWS Ground Station has a service quota on the number of `ENABLED` customer-provided ephemerides per satellite (see: Service Quotas). To upload ephemeris data after reaching this quota, delete (using `DeleteEphemeris`) or disable (using `UpdateEphemeris`) the lowest-priority/earliest created customer-provided ephemerides.

If no ephemeris has been created, or if no ephemerides have `ENABLED` status, AWS Ground Station will use a default ephemeris for the satellite (from Space Track), if available. This default ephemeris has priority 0.

Effect of new Ephemerides on Previously Scheduled Contacts

Use the DescribeContact API to view the effects of new ephemerides on previously scheduled contacts by returning the active visibility times.

Contacts scheduled prior to uploading a new ephemeris will retain the originally scheduled contact time, while the antenna tracking will use the active ephemeris. If the spacecraft's position, based on the active ephemeris, differs greatly from the prior ephemeris, this may result in reduced satellite contact time with the antenna due to the spacecraft operating outside the transmit/ receive site mask. Therefore, we recommend that you cancel and reschedule your future contacts after you upload a new ephemeris that differs greatly from the previous ephemeris. With the DescribeContact API, you can determine the portion of your future contact that is unusable due to the spacecraft operating outside the transmit/receive site mask by comparing your scheduled contact `startTime` and `endTime` with the returned `visibilityStartTime` and `visibilityEndTime`. If you choose to cancel and reschedule your future contact(s), the contact time range must not be outside the visibility time range by more than 30 seconds. Cancelled contacts may incur costs when cancelled too close to the time of contact. For more information on cancelled contacts see: Ground Station FAQs.

Getting the Current Ephemeris for a Satellite

The current ephemeris in use by AWS Ground Station for a specific satellite can be retrieved by calling the `GetSatellite` or `ListSatellites` actions. Both of these methods will return metadata for the ephemeris currently in use. This ephemeris metadata is different for custom ephemerides uploaded to AWS Ground Station and for default ephemerides.

Default Ephemerides will only include `source` and `epoch` fields. The `epoch` is the [epoch](#) of the [two-line element set](#) that was pulled from Space Track AWS Ground Station, and it is currently being used for computing the trajectory of the satellite.

A custom ephemeris will have a `source` value of `"CUSTOMER_PROVIDED"` and will include a unique identifier in the `ephemerisId` field. This unique identifier can be used to query for the ephemeris via the `DescribeEphemeris` action. An optional `name` field will be returned if the ephemeris was assigned a name during upload to AWS Ground Station via the `CreateEphemeris` action.

It is important to note that ephemerides are updated dynamically by AWS Ground Station so the returned data is only a snapshot of the ephemeris being used at the time of the call to the API.

Example `GetSatellite` return for a satellite using a default ephemeris

```
{
    "satelliteId": "e1cfe0c7-67f9-4d98-bad2-06dbfc2d14a2",
    "satelliteArn": "arn:aws:groundstation::111122223333:satellite/e1cfe0c7-67f9-4d98-
bad2-06dbfc2d14a2",
    "noradSatelliteID": 12345,
    "groundStations": [
        "Example Ground Station 1",
        "Example Ground Station 2"
    ],
    "currentEphemeris": {
        "source": "SPACE_TRACK",
        "epoch": 8888888888
    }
}
```

Example `GetSatellite` for a satellite using a custom ephemeris

```
{
      "satelliteId": "e1cfe0c7-67f9-4d98-bad2-06dbfc2d14a2",
      "satelliteArn": "arn:aws:groundstation::111122223333:satellite/
e1cfe0c7-67f9-4d98-bad2-06dbfc2d14a2",
      "noradSatelliteID": 12345,
      "groundStations": [
          "Example Ground Station 1",
          "Example Ground Station 2"
      ],
      "currentEphemeris": {
          "source": "CUSTOMER_PROVIDED",
          "ephemerisId": "e1cfe0c7-67f9-4d98-bad2-06dbfc2d14a2",
          "name": "My Ephemeris"
      }
  }
```

Providing Custom Ephemeris Data

Warning

The ephemeris API is currently in a Preview state

Access to the Ephemeris API is provided only on an as-needed basis. Customers requiring the ability to upload custom ephemeris data should contact aws-groundstation@amazon.com.

Overview

The Ephemeris API allows custom ephemerides to be uploaded to AWS Ground Station for use with a satellite. These ephemerides override the default ephemerides from Space Track (see: Default Ephemeris Data).

Uploading customer ephemerides can improve the quality of tracking, handle early operations where no Space Track ephemerides are available to AWS Ground Station, and to account for maneuvers.

Creating a custom Ephemeris

A custom ephemeris can be created using the `CreateEphemeris` action in the AWS Ground Station API. This action will upload an ephemeris using data either in the request body or from a specified S3 bucket.

It is important to note that uploading an ephemeris sets the ephemeris to `VALIDATING` and starts an asynchronous workflow that will validate and generate potential contacts from your ephemeris. Only once an ephemeris has passed this workflow and become `ENABLED` will it be used for contacts. You should poll `DescribeEphemeris` for the ephemeris status or use Cloudwatch events to track the ephemeris' status changes.

To troubleshoot an invalid ephemeris see: Troubleshooting Invalid Ephemerides

Create a TLE Set Ephemeris via API

The AWS Ground Station boto3 client can be used to upload a two line element (TLE) set ephemeris to AWS Ground Station via the `CreateEphemeris` call. This ephemeris will be used in place of the default ephemeris data for a satellite (see Default Ephemeris Data).

A TLE set is a JSON formatted object that strings one or more TLEs together to construct a continuous trajectory. The TLEs in the TLE set must form a continuous set that we can use to construct a trajectory (i.e. no gaps in time between TLEs in a TLE set). An example TLE set is shown below:

```
# example_tle_set.json
[
    {
        "tleLine1": "1 25994U 99068A   20318.54719794  .00000075  00000-0  26688-4 0
 9997",
        "tleLine2": "2 25994  98.2007  30.6589 0001234  89.2782  18.9934
 14.57114995111906",
        "validTimeRange": {
            "startTime": 12345,
            "endTime": 12346
        }
    },
    {
        "tleLine1": "1 25994U 99068A   20318.54719794  .00000075  00000-0  26688-4 0
 9997",
        "tleLine2": "2 25994  98.2007  30.6589 0001234  89.2782  18.9934
 14.57114995111906",
        "validTimeRange": {
            "startTime": 12346,
            "endTime": 12347
        }
    }
]
```

Note

The time ranges of the TLEs in a TLE set must match up exactly to be a valid, continuous trajectory.

A TLE set can be uploaded via the AWS Ground Station boto3 client as follows:

```
tle_ephemeris_id = ground_station_boto3_client.create_ephemeris( name="Example
 Ephemeris", satelliteId="2e925701-9485-4644-b031-EXAMPLE01", enabled=True,
 expirationTime=datetime.now(timezone.utc) + timedelta(days=3), priority=2,
    ephemeris = {
      "tle": {
        "tleData": [
            {
                "tleLine1": "1 25994U 99068A   20318.54719794  .00000075  00000-0
 26688-4 0  9997",
                "tleLine2": "2 25994  98.2007  30.6589 0001234  89.2782  18.9934
 14.57114995111906",
                "validTimeRange": {
                    "startTime": datetime.now(timezone.utc),
                    "endTime": datetime.now(timezone.utc) + timedelta(days=7)
                }
            }
        ]
      }
    })
```

This call will return an ephemeris Id that can be used to reference the ephemeris in the future. For example, we can use the provided ephemeris ID from the call above to poll for the status of the ephemeris:

```
client.describe_ephemeris(ephemerisId=tle_ephemeris_id['ephemerisId'])
```

An example response from the `DescribeEphemeris` action is provided below

```
{
  "creationTime": 1620254718.765,
  "enabled": true,
  "name": "Example Ephemeris",
  "ephemerisId": "fde41049-14f7-413e-bd7b-EXAMPLE01",
  "priority": 2,
  "status": "VALIDATING",
  "suppliedData": {
    "tle": {
      "ephemerisData": "[{\"tleLine1\": \"1 25994U 99068A   20318.54719794  .00000075
 00000-0  26688-4 0  9997\",\"tleLine2\": \"2 25994  98.2007  30.6589 0001234  89.2782
  18.9934 14.57114995111906\",\"validTimeRange\": {\"startTime\": 1620254712000,
\"endTime\": 1620859512000}}]"
    }
  }
}
```

It is recommended to poll the `DescribeEphemeris` route or use Cloudwatch events to track the status of the uploaded ephemeris as it must go through an asynchronous validation workflow before it is set to `ENABLED` and becomes usable for scheduling and executing contacts.

Note that the NORAD ID in all TLEs in the TLE set, `25994` in the examples above, must match the NORAD ID your satellite has been assigned in the Space Track database.

Uploading Ephemeris data from an S3 bucket

It is also possible to upload an ephemeris file directly from an S3 bucket by pointing to the bucket and object key. AWS Ground Station will retrieve the object on your behalf. Information about the encryption of data at rest in AWS Ground Station is detailed in: [Data Encryption At Rest For AWS Ground Station]

Below is an example of uploading an OEM ephemeris file from an S3 bucket

```
s3_oem_ephemeris_id = customer_client.create_ephemeris( name="2022-10-26 S3
 OEM Upload", satelliteId="fde41049-14f7-413e-bd7b-EXAMPLE01", enabled=True,
 expirationTime=datetime.now(timezone.utc) + timedelta(days=5), priority=2,
    ephemeris = {
        "oem": {
            "s3Object": {
                "bucket": "ephemeris-bucket-for-testing",
                "key": "test_data.oem",
            }
        }
    })
```

Below is an example returned data from the `DescribeEphemeris` action being called for the OEM ephemeris uploaded in the previous block of example code.

```
{
    "creationTime": 1620254718.765,
    "enabled": true,
    "name": "Example Ephemeris",
    "ephemerisId": "fde41049-14f7-413e-bd7b-EXAMPLE02",
    "priority": 2,
    "status": "VALIDATING",
    "suppliedData": {
    "oem": {
        "sourceS3Object": {
            "bucket": "ephemeris-bucket-for-testing",
            "key": "test_data.oem"
        }
    }
}
```

Troubleshooting Invalid Ephemerides

When a custom ephemeris is uploaded to AWS Ground Station it goes through an asynchronous validation workflow before becoming `ENABLED`. This workflow ensures that the satellite identifiers, metadata, and trajectory are valid.

When an Ephemeris fails validation, `DescribeEphemeris` will return an *EphemerisInvalidReason*, which provides insight into why the ephemeris failed validation. The potential values of the *EphemerisInvalidReason* are as follows:

Value	Description	Troubleshooting Action
METADATA_INVALID	Provided spacecraft identifiers such as satellite ID are invalid	Check the NORAD ID or other identifiers provided in the ephemeris data
TIME_RANGE_INVALID	Start, end, or expiration time(s) are invalid for the provided ephemeris	Make sure the Start time is before `now` (it is recommended to set the start time a few minutes in the past), that the end time

Value	Description	Troubleshooting Action
		is after the start time, and that the end time is after the expiration time
TRAJECTORY_INVALID	Provided ephemeris defines an invalid spacecraft trajectory	Confirm that the provided trajectory is continuous and is for the correct satellite.
VALIDATION_ERROR	Internal service error occurred while processing ephemeris for validation	Retry Upload

An example `DescribeEphemeris` response for an `INVALID` ephemeris is provided below:

```
{
  "creationTime": 1000000000.00,
  "enabled": false,
  "ephemerisId": "d5a8a6ac-8a3a-444e-927e-EXAMPLE1",
  "name": "Example",
  "priority": 2,
  "status": "INVALID",
  "invalidReason": "METADATA_INVALID",
  "suppliedData": {
    "tle": {
      "sourceS3Object": {
        "bucket": "my-s3-bucket",
        "key": "myEphemerisKey",
        "version": "ephemerisVersion"
      }
    }
  },
}
```

Reverting To Default Ephemeris Data

When you upload custom ephemeris data it will override the default ephemerides AWS Ground Station uses for that particular satellite. AWS Ground Station does not use the default ephemeris again until there are no currently enabled, unexpired customer-provided ephemerides available for use. AWS Ground Station also does not list contacts past the expiration time of the current customer-provided ephemeris, even if there is a default ephemeris available past that expiration time.

To revert back to the default Space Track ephemerides, you will need to do one of the following:

- Delete (using `DeleteEphemeris`) or disable (using `UpdateEphemeris`) all enabled customer-provided ephemerides. You can list the customer-provided ephemerides for a satellite using `ListEphemerides`.
- Wait for all existing customer-provided ephemerides to expire.

You can confirm that the default ephemeris is being used by calling `GetSatellite` and verifying that the `source` of the current ephemeris for the satellite is `SPACE_TRACK`. See Default Ephemeris Data for more information on default ephemerides.

AWS Ground Station Site Masks

Each AWS Ground Station antenna location has associated site masks. These masks block antennas at that location from transmitting or receiving when pointing in some directions, typically close to the horizon. The masks may take into account:

- **Features of the geographic terrain surrounding the antenna.** For example, this includes things like mountains or buildings, that would block a radio frequency (RF) signal or prevent transmitting.
- **Radio Frequency Interference (RFI)** This affects both the ability to receive (external RFI sources impacting a downlink signal into AWS Ground Station antennas) and transmit (the RF signal transmitted by AWS Ground Station antennas adversely impacting external receivers).
- **Legal authorizations.** Local site authorizations to operate AWS Ground Station in each region may include specific restrictions, such as a minimum elevation angle for transmitting.

These site masks may change over time. For example, new buildings could be constructed near an antenna location, RFI sources may change, or legal authorization may be renewed with different restrictions. The AWS Ground Station site masks are available to customers under a non-disclosure agreement (NDA).

Customer-Specific Masks

In addition to the AWS Ground Station site masks at each site, each customer may have additional masks due to restrictions on their own legal authorization to communicate with their satellites in a given region. Such masks can be configured in AWS Ground Station on a case-by-case basis to ensure compliance when using AWS Ground Station to communicate with these satellites. Contact the AWS Ground Station team for details.

Impact of Site Masks on Available Contact Times

There are two kinds of site masks: uplink (transmit) site masks, and downlink (receive) site masks.

When listing available contact times using the ListContacts operation, AWS Ground Station will return visibility times based on when your satellite will rise above and set below the downlink mask. Available contact times are based on this downlink mask visibility window. This ensures that customers do not reserve or pay for time when their satellite is below the downlink mask.

Uplink site masks are *not* applied to the available contact times, even if the Mission Profile includes an Antenna Uplink Config in a dataflow edge. This allows customers to use all available contact time for downlink, even if uplink may not be available for portions of that time due to the uplink site mask. However, the uplink signal may not be transmitted for some or all of the time reserved for a satellite contact. Customers are responsible for accounting for the provided uplink mask when scheduling uplink transmissions.

The portion of a contact that is unavailable for uplink varies depending on the satellite trajectory during the contact, relative to the uplink site mask at the antenna location. In regions where the uplink and downlink site masks are similar, this duration will typically be short. In other regions, where the uplink mask may be considerably higher than the downlink site mask, this could result in significant portions, or even all, of the contact duration being unavailable for uplink. The full contact time is billed to the customer, even if portions of the reserved time are unavailable for uplink.

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.

Change	Description	Release Date
New Feature	Contacts can now be scheduled up to 30 seconds outside visibility time ranges. Visibility times are included in DescribeContact responses.	Mar 26, 2024
Documentation Update	Improved organization and added "EC2 Instance Selection and CPU Planning" section.	Mar 6, 2024
Documentation Update	Added new best practice to AWS Ground Station Agent User Guide for running services and processes alongside the AWS Ground Station Agent.	Feb 23, 2024
Documentation Update	Added Agent Release Notes page.	Feb 21, 2024
Template Update	Added support for separate public subnet in the DirectBroadcastSatelliteWbDigIfEc2DataDelivery template.	Feb 14, 2024
Documentation Update	Added referral to AWS User Notifications in monitoring documentation.	Aug 6, 2023
Documentation Update	Added instructions for tagging satellites with a name to be shown in the AWS Ground Station console.	July 26, 2023

Change	Description	Release Date
New Feature	Added the AWS Ground Station Agent User Guide for the release of Wideband DigIF Data Delivery	April 12, 2023
AWS managed policy updates – New AWS managed policy	AWS Ground Station added a new policy named AWSGroundStationAgentInstancePolicy.	April 12, 2023
New Feature	Updated the user guide for release of CPE Preview.	November 9, 2022
AWS managed policy updates – New AWS managed policy	AWS Ground Station added the AWSServiceRoleForGroundStationDataflowEndpointGroup service-linked-role (SLR) that includes a new policy named AWSServiceRoleForGroundStationDataflowEndpointGroupPolicy.	November 02, 2022
New Feature	Updated the user guide to include integration with AWS CLI.	April 17, 2020
New Feature	Updated the user guide to include integration with CloudWatch Metrics.	February 24, 2020
New Template	Public Broadcast Satellites (AquaSnppJpss Template) added to the *AWS Ground Station User Guide*.	February 19, 2020
New Feature	Updated the user guide to include cross-region data delivery.	February 5, 2020
Documentation Update	Updated examples and descriptions for monitoring AWS Ground Station with CloudWatch Events.	February 4, 2020
Documentation Update	Template locations have been updated and the Getting Started and Troubleshooting sections have been revised.	December 19, 2019

Change	Description	Release Date
New Troubleshooting Section	Troubleshooting section added to the *AWS Ground Station User Guide*.	November 7, 2019
New Getting Started Topic	Updated the Getting Started topic, which includes the most current AWS CloudForm ation templates.	July 1, 2019
Kindle Version	Published Kindle version of the *AWS Ground Station User Guide*.	June 20, 2019
New service and guide	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 Glossary Reference*.