AWS Ground Station Agent User Guide

AWS Ground Station

AWS Ground Station: AWS Ground Station Agent User Guide

Table of Contents

Overview

What is the AWS Ground Station Agent?

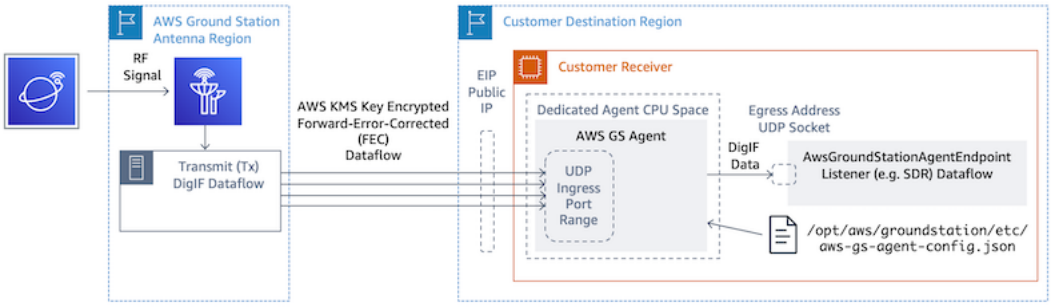
With the AWS Ground Station Agent, available as an RPM, you can receive (downlink) synchronous Wideband Digital Intermediate Frequency (DigIF) dataflows during AWS Ground Station contacts. You can select two options for data delivery:

1. **Data delivery to an EC2 instance** - Data delivery to an EC2 instance that you own. You 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 Elastic Compute Cloud guide for information about EC2 data delivery.
2. **Data delivery to an S3 bucket** - Data delivery to an AWS S3 bucket that you own 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 you to create a set of AWS resources. The use of CloudFormation 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).

The following diagram shows a 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 AWS Ground Station Getting started 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 in the same availability zone as the public 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.

For example AWS CloudFormation EC2 Data Delivery templates with these resources preconfigured, see Public broadcast satellite utilizing AWS Ground Station Agent (wideband) .

VPC diagrams

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

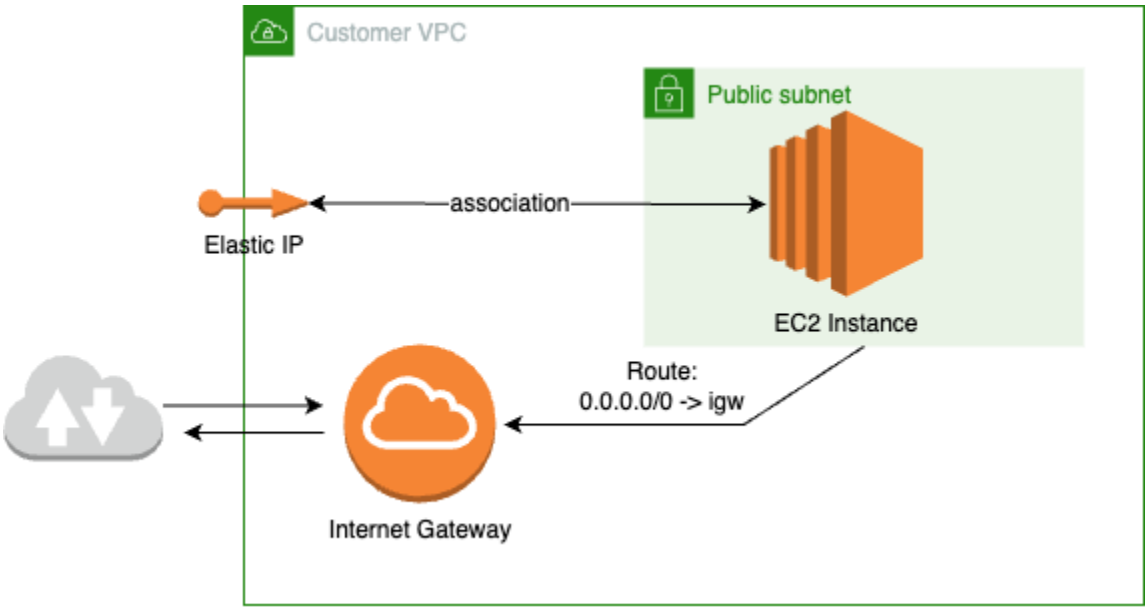
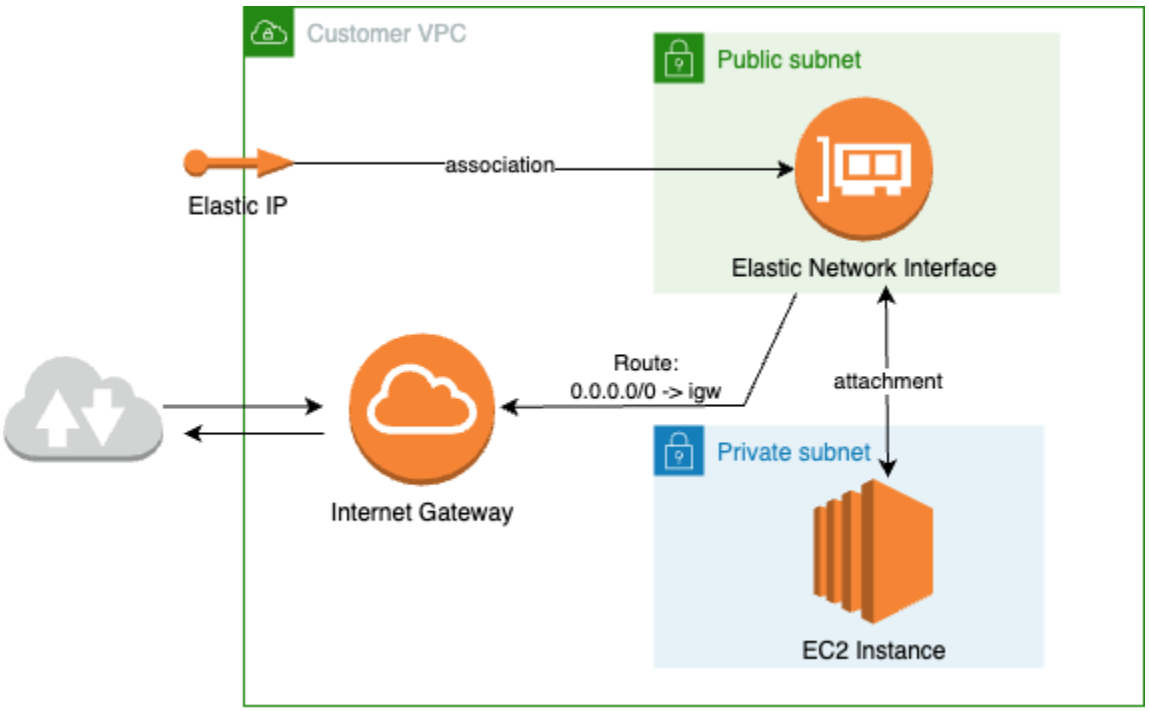


Diagram: An Elastic IP associated with an ENI in a public subnet, attached to your EC2 instance in a private subnet



Supported operating system

Amazon Linux 2 with 5.10+ kernel.

Supported instances types are listed in Select Amazon EC2 instance and reserve CPU cores for your architecture

Receive data via the 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.

You 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 you prefer to receive all dataflows on a single receiver instance.

Multiple dataflows, single receiver

Example Scenario:

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

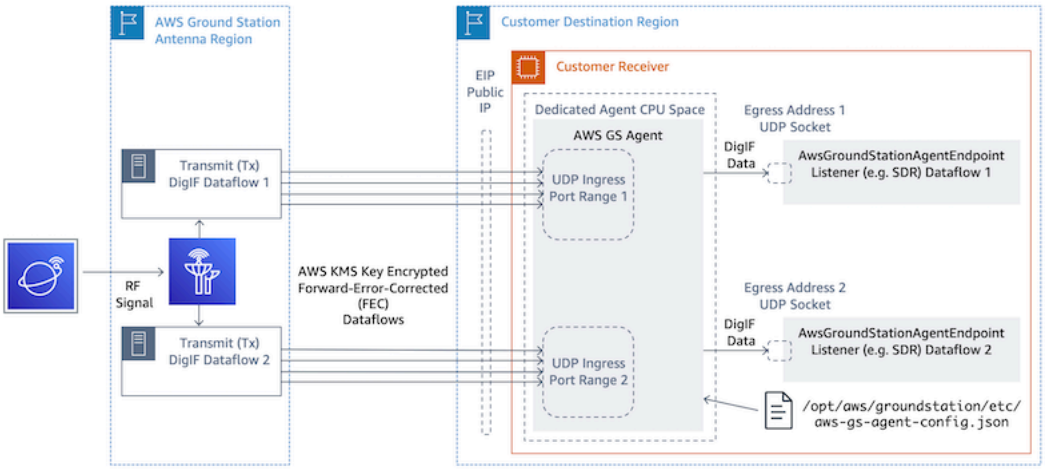
AwsGroundStationAgentEndpoints:

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.



Multiple dataflows, multiple receivers

Example Scenario:

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

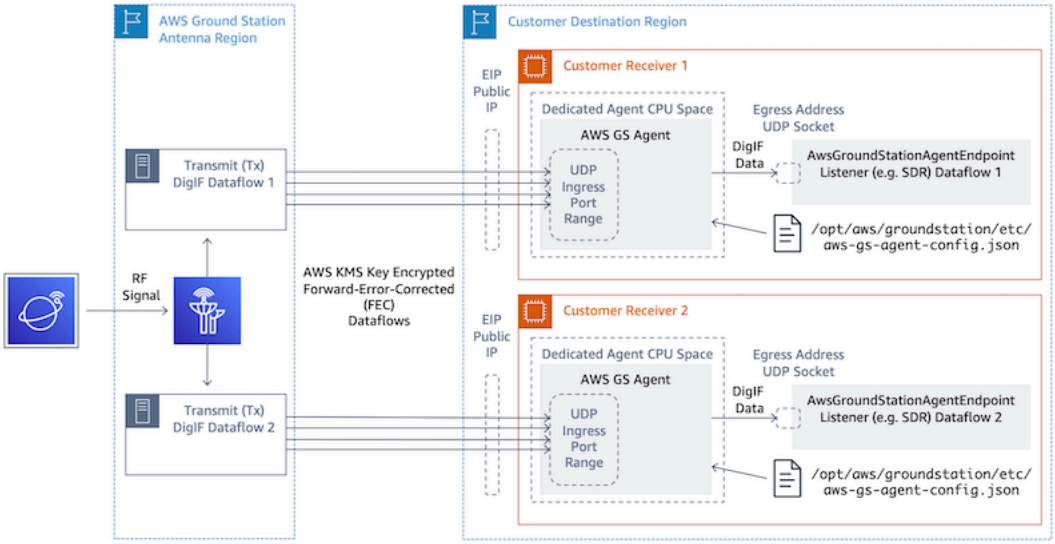
AwsGroundStationAgentEndpoints:

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.



Select Amazon EC2 instance and reserve CPU cores for your architecture

Supported Amazon 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

Instance type	Default vCPUs	Default CPU cores
p4d.24xlarge	96	48
r5.24xlarge	96	48
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

AntennaDownlink Bandwidth (MHz)	Expected VITA-49.2 DigIF Data Rate (Mb/s)	Number of Cores (HT CPU Pairs)	Total vCPU
200	4000	6	12
250	5000	6	12
300	6000	7	14
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 50-66 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, 50, 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
```

Install 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.

Use a AWS CloudFormation template

The EC2 data delivery AWS 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 (Tune your EC2 instance for performance).

Step 1: Create AWS resources

Create your AWS resources stack using template Public broadcast satellite utilizing AWS Ground Station Agent (wideband) .

Step 2: Check agent status

By default 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.

Install manually 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 Example mission profile configurations 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 you. You 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 Configure 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 Tune your EC2 instance for performance and Reboot the instance.

Step 6: Manage the agent

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

Manage the agent

The 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

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
```

Configure 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"
]
```

Tune your EC2 instance for performance

> **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

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 that's 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 that's 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

Family	Instance Type	${interrupt_core_list}	${rps_core_mask}
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 • 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

Family	Instance Type	${interrupt_core_list}	${rps_core_mask}
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 Tune your EC2 instance for performance.
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

Amazon EC2 best 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.

Document history for the AWS Ground Station Agent User Guide

The following table describes the important changes in each release of the AWS Ground Station Agent User Guide.

Change	Description	Date
Documentation Update	Added comment about keeping the subnet and the Amazon EC2 instance in the same availability zone in Agent Requirements.	July 18, 2024
Documentation Update	Split the AWS Ground Station Agent into its own user guide. For prior changes, please refer to: Document history for the AWS Ground Station user guide.	July 18, 2024