opentext™

# ArcSight SmartConnectors

Software Version: 24.2

## Configuration Guide for Amazon Web Services Cloudwatch SmartConnector

Document Release Date: April 2024
Software Release Date: April 2024

# Legal Notices

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

# Copyright Notice



# Trademark Notices



# Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number
- Document Release Date, which changes each time the document is updated
- Software Release Date, which indicates the release date of this version of the software

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

https://www.microfocus.com/support-and-services/documentation

# Contents

# Configuration Guide for Amazon Web Services CloudWatch SmartConnector

Amazon Web Services CloudWatch (AWS CloudWatch ) is a Cloud-native Connector that is deployed on the cloud environment.

AWS CloudWatch SmartConnector helps you to gather all the event logs generated inside a specific Virtual Private Cloud (VPC), normalizes event data to Common Event Format (CEF), and sends the data to a configured ArcSight destination.

### Intended Audience

This guide provides information for IT administrators who are responsible for managing the ArcSight software and its environment.

### Additional Documentation

The ArcSight SmartConnector documentation library includes the following resources:

- Technical Requirements Guide for SmartConnector, which provides information about operating system, appliance, browser, and other support details for SmartConnector.
- Installation and User Guide for SmartConnectors, which provides detailed information about installing SmartConnectors.
- Configuration Guides for ArcSight SmartConnectors, which provides information about configuring SmartConnectors to collect events from different sources.
- Configuration Guide for SmartConnector Load Balancer, which provides detailed information about installing Load Balancer.

For the most recent version of this guide and other ArcSight SmartConnector documentation resources, visit the documentation site for ArcSight SmartConnectors.

### Contact Information

We want to hear your comments and suggestions about this book and the other documentation included with this product. You can use the comment on this topic link at the bottom of each page of the online documentation, or send an email to MFI-Documentation-Feedback@opentext.com.

For specific product issues, contact Open Text Support for Micro Focus products.

# Product Overview

Amazon CloudWatch events are a stream of system events that describe changes in AWS resources. You can use simple rules that you can quickly set up, match events, and route those events to one or more target functions or streams.

Amazon CloudWatch helps and monitors the AWS resources and the applications you run on AWS in real time. AWS CloudWatch SmartConnector allows you to use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

# Supported Log Sources

AWS CloudWatch Connector collects events from the following log sources:

- **VPC Flow Logs**

  VPC Flow Logs in Amazon Virtual Private Cloud (Amazon VPC) enables you to capture information about network or IP traffic going around network interfaces within your VPC. VPC flow logs indicate a change in your AWS environment.

  AWS resources generate events when their state changes. For example, an Amazon EC2 instance generates events when the state of an EC2 changes from **Pending** to **Running**, EC2 starts or stops, EC2 gets terminated, Amazon EC2 Auto Scaling generates events when it launches or terminates instances, etc.

- **CloudHSM Audit Logs**

  CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use your own encryption keys on AWS Cloud. CloudHSM is standards-compliant and enables you to export all your encryption keys to the other commercially-available HSMs, subject to your configurations.

  The CloudHSM audit logs include all client-initiated management commands, including those that create and delete the HSM, log in to and out of HSM, and manage users and keys. These logs provide a reliable record of actions that have changed the state of HSM.

  AWS CloudHSM collects your HSM audit logs and sends them to Amazon CloudWatch Logs.

- **Route 53 Logs**

  Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. You can use Amazon Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints. Amazon Route 53 Traffic Flow makes it easy for you to manage the traffic globally through a variety of routing types, including Latency Based Routing, Geo DNS, Geoproximity, and Weighted Round Robin—all of which can be combined with DNS Failover to enable a variety of low-latency, fault-tolerant architectures.

# Related AWS Services

The following services are used in conjunction with CloudWatch Events:

- **AWS CloudFormation**

  It enables you to model and set up your AWS resources. AWS CloudFormation takes care of provisioning and configuring AWS resources based on a template describing your requirements. You can use CloudWatch Events rules in your AWS CloudFormation templates. For more information, see the AWS documentation.

- **AWS Identity and Access Management (IAM)**

  It helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication), what resources they can use, and how they can use them (authorization). For more information, see the AWS documentation.

- **Amazon Kinesis Data Streams**

  It enables rapid and nearly continuous data intake and aggregation. The type of data used includes IT infrastructure log data, application logs, social media, market data feeds, and web clickstream data. Because the response time for the data intake and processing is in real time, processing is typically lightweight. For more information, see the AWS documentation.

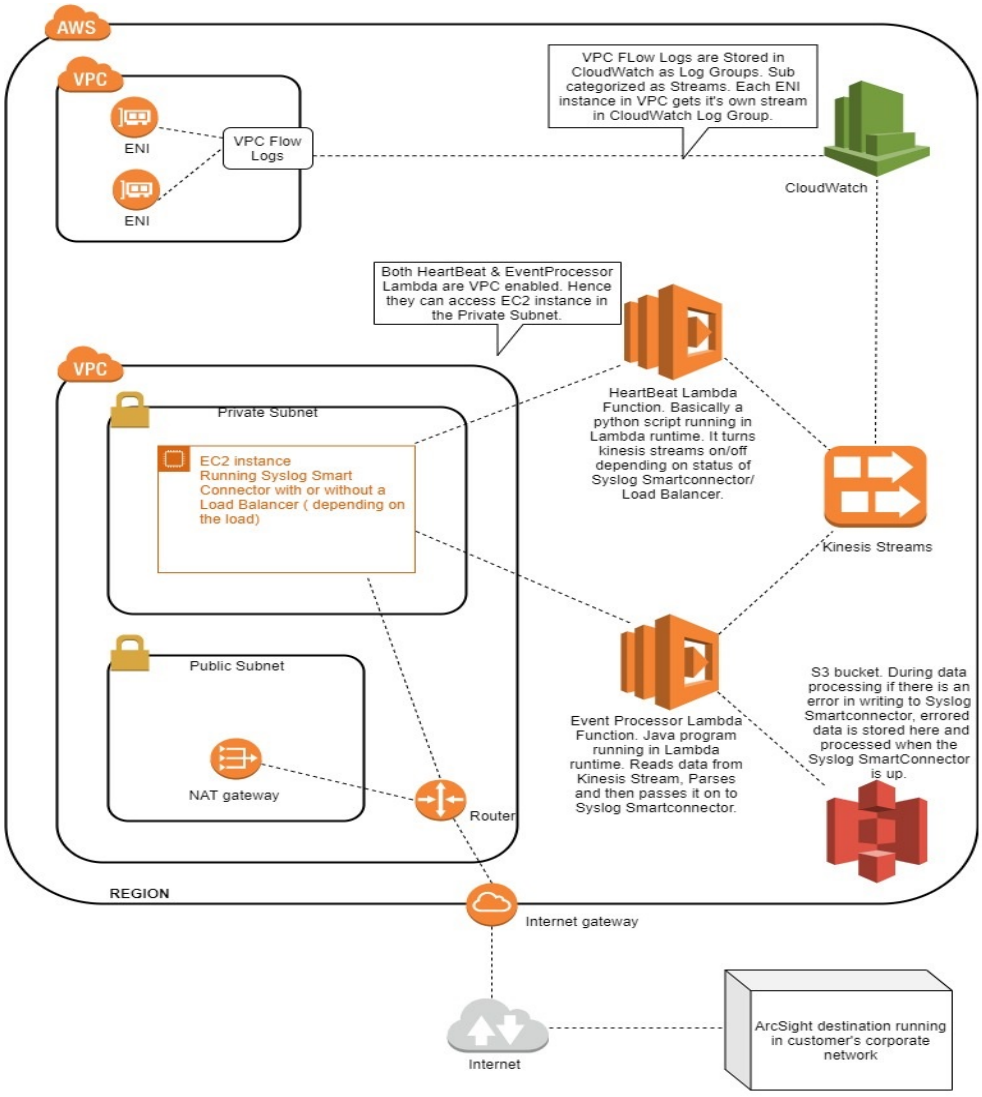
- **AWS Lambda**

  It enables you to build applications that respond quickly to new information. Upload your application code as Lambda functions and Lambda runs your code on high-availability compute infrastructure. Lambda performs all the administration of the compute resources, including server and operating system maintenance, capacity provisioning, automatic scaling, code and security patch deployment, and code monitoring and logging. For more information, see the AWS documentation.

# Understanding Data Collection

The following diagram provides a high-level overview of how the AWS CloudWatch Connector collects and sends VPC Flow Logs, Route 53 logs and CloudHSM audit logs to a configured ArcSight destination.

**Note**: The diagram shows only VPC Flow Logs. However, note that the AWS CloudWatch Connector also supports Route 53 and CloudHSM audit logs.




The following steps describe an overview of the AWS CloudWatch SmartConnector design to understand the data collection flow:

1. After you have configured VPC Flow, CloudHSM, and Route 53 as log sources, the events will be displayed in the CloudWatch Log Groups. You can access these Log Groups from the CloudWatch Logs dashboard. Each Log Group contains a separate stream for each Elastic Network Interface (ENI) instance.
2. The events will be flowing from the CloudWatch Logs log groups to Kinesis Data Stream (subscription).
3. The processing in the design is performed by Lambda Monitoring.

   Lambda Monitoring will continuously monitor the status of the destination that is Syslog NG Daemon SmartConnector (with or without Load Balancer).

   Lambda Monitoring enables the Kinesis Data Stream subscription to the Lambda Event Processing function based on the accessibility of the destination:

   - If the configured destination (Syslog NG Daemon SmartConnector) is up and available, events will flow from Kinesis Stream to Lambda Event Processor for parsing. It decodes the batch, loads it into the memory in plain text, and then parses and processes the events contained in that object.
   - If the configured destination (Syslog NG Daemon SmartConnector) is down, Lambda Event Processor function will back up the events to an S3 bucket.
4. After the events are parsed, they are sent to the configured ArcSight destination such as ArcSight Logger, ESM, or CEF file.

# Configuring Log Sources

## Configuring Route 53

You can configure **Amazon Route 53** to log information about queries such as :

- Requested main domains or subdomains
- Date and time of a request
- DNS record types (such as A or AAAA)
- Route 53 edge locations responding to a DNS query
- A DNS response code, such as **NoError** or **ServFail**

Amazon Route 53 sends query logs directly to CloudWatch Logs. The logs are not accessible through Route 53. Instead, you use CloudWatch Logs to view logs in near real-time, search and filter data, and export logs to Amazon S3.

## Prerequisites

- One or more public domains

## Query Logging

Query logs contain queries that DNS resolvers forward to Route 53. If a DNS resolver caches a response of a query, for example, the IP address of a Load Balancer, the resolver continues to return the cached response without forwarding the query to Route 53, until the time to live (TTL) of the record expires.

Query logs might contain the information of one single query, out of every several thousand queries submitted to DNS resolvers. This depends on the DNS queries submitted from one single domain name or a subdomain name, the resolvers being used, and the TTL of the record. For more information about how DNS works, see the AWS Documentation.

Following is an example of a Route 53 Query Log:

```
1.0 2017-12-13T08:15:50.235Z Z123412341234 example.com AAAA NOERROR TCP IAD12
192.168.3.1 192.168.222.0/24

1.0 2017-12-13T08:16:03.983Z Z123412341234 example.com ANY NOERROR UDP FRA6
2001:db8::1234 2001:db8:abcd::/48

1.0 2017-12-13T08:15:50.342Z Z123412341234 bad.example.com A NXDOMAIN UDP
IAD12 192.168.3.1 192.168.111.0/24

1.0 2017-12-13T08:16:05.744Z Z123412341234 txt.example.com TXT NOERROR UDP
JFK5 192.168.1.2 -
```

## Using Amazon CloudWatch to Access DNS Query Logs

Route 53 creates one CloudWatch Logs log stream for each Route 53 edge location that responds to DNS queries of the specified hosted zone and sends query logs to the applicable log stream.

The format of each log stream name is hosted-zone-id/edge-location-ID, for example, `Z1D633PJN98FT9/DFW3`.

Each edge location is identified by a three-letter code and an arbitrarily assigned number, for example, DFW3. The three-letter code typically corresponds with the International Air Transport Association airport code for an airport near the edge location. For a list of edge locations, see Route 53 Product Details.

# Configuring Query Logging

1. Log in to the AWS Management Console and open the Route 53 console: https://console.aws.amazon.com/route53/
2. From **Navigation**, click **Hosted Zones**.
3. Select the hosted zone you are configuring.
4. In **Hosted Zone Details**, select **Configure Query Logging**.
5. From **Send Query Logs to**, select one of the following options:
   a. New Log Group in US East (North Virginia)
   b. Existing Log Group in US East (North Virginia)

   If you want to configure this feature for multiple hosted zones, we recommend you to use a consistent prefix for every log group, for example: `/aws/route53/hosted-zone-name`. Resource policies grant permissions to Route 53 and let users publish logs to the Log Group. The maximum number of resource policies that you can create for an AWS account is 10. And, by using a consistent prefix in your Log Group names, you can choose one resource policy for all your Route 53 hosted zones.

   **Note:** Log Groups must be in the US East (North Virginia) Region.
6. If choosing, **Existing Log Group in US East (North Virginia)**, enter the **Existing Log Group Name** and click **Next**. If not, go to step 7.

   Route 53 can use the same permissions in any existing Resource Policy, you do not need to choose a specific Resource Policy.
   a. Click **Test** to determine if the existing resource policies grant the permissions required for Route 53 to publish logs to the log group. If the test fails, users may follow one of these options:
      i. Continue to step 7, to create a New Resource Policy, or:
      ii. Click **Edit** and continue to change the value **Log groups that the resource policy applies to**. Specify the name of a CloudWatch Logs log group, for example /aws/route53/example.com, or a value that includes the current log group, like, /aws/route53/*. Next, move on to step 8.

      **Note:** Users can use the wildcard character (*) to replace 0 or to more characters in the name of the log group.
7. To create a **New Log Group in US East (North Virginia)**:

a. Enter the **Resource Policy Name**.

b. Enter a valid value in **Log groups that the resource policy applies to** .

   The name of the Log Group is at the top of the current page.

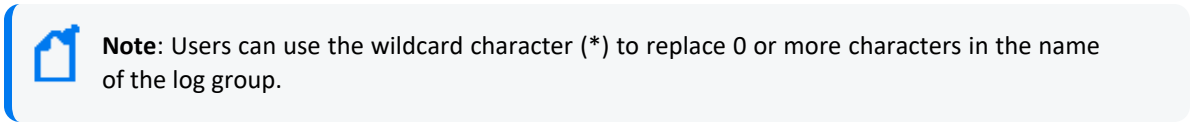
c. Click **Create Policy and test permissions** to determine if the existing resource policies grant the permissions required for Route 53 to publish logs to the log group. If the error "Resource limit exceeded" pops:

   i. Click **Edit** in one of the existing resource policies.

   ii. Change the value **Log groups that the resource policy applies to**. Specify the name of a log group, for example `/aws/route53/example.com`, or a value that includes the current log group, like, `/aws/route53/*`.

   To view the settings of a Resource Policy, click the arrow on the left side of the resource policy name.

   iii. Click **Save policy and test permissions**.




8. Select **Create Query Logging Config**.

## Stopping Query Logging

To stop query logging so that Amazon Route 53 no longer sends query logs to CloudWatch Logs:

1. Log in to the **AWS Management Console**.
2. Open the Route 53 Console.
3. From **Navigation**, click **Hosted zones**.
4. Select the hosted zone from where you are deleting the query logging.
5. From **Hosted Zone Details** > **Query Logging,** click **Delete**.
6. Click **Confirm** to delete the query logging configuration.

For more information, see AWS documentation.

# Configuring AWS CloudHSM for Event Colletion

AWS CloudHSM provides hardware security modules in the AWS Cloud. A hardware security module (HSM) is a computing device that processes cryptographic operations and provides secure storage for cryptographic keys.

# Configuring AWS Cloud HSM

**To Configure AWS CloudHSM:**

1. Create IAM Administrative Groups.
2. Create a Virtual Private Cloud (VPC).
3. Create a Private Subnet.
4. Create a Cluster.
5. Review Cluster Security Group.
6. Launch an Amazon EC2 Client Instance.
7. Connect Amazon EC2 Instance to AWS CloudHSM Cluster.
8. Create an HSM.
9. Verify the Identity and Authenticity of Your Cluster's HSM (Optional).
10. Initialize the Cluster.
11. Install and Configure the AWS CloudHSM Client (Linux)Install and Configure the AWS CloudHSM Client (Windows).
12. Activate the Cluster.

For more information, see Getting Started with AWS CloudHSM.

# Configuring AWS CloudHSM Audit Logs

AWS CloudHSM provides command line tools to manage and use AWS CloudHSM.

**Managing Clusters and HSMs**

The following tools get, create, delete, and tag AWS CloudHSM clusters and HSMs:

- cloudhsmv2. To use these commands, you need to install and configure AWS CLI.
- HSM2 PowerShell cmdlets in the AWS Tools for PowerShell. These cmdlets are available in a Windows PowerShell module and a cross-platform PowerShell Core module.

**Managing Users**

The following tool creates and deletes HSM users, including implementing quorum authentication of user management tasks:

The **cloudhsm_mgmt_util** command line tool helps crypto officers manage users in the HSMs. It also includes commands that allow crypto users (CUs) to share keys and get and set key attributes. It consists of tools that create, delete, and list users, and change user passwords. This tool is included in the Install the AWS CloudHSM Client and Command Line Tools.

## Preparing to run cloudhsm_mgmt_util

Complete the following tasks before you use cloudhsm_mgmt_util the first time you use cloudhsm_mgmt_util and after you add or remove HSMs in your cluster.

1. **Stop the AWS CloudHSM Client**
   - Amazon Linux, CentOS 6, RHEL 6: $ sudo stop cloudhsm-client
   - Amazon Linux 2, CentOS 7, RHEL 7, Ubuntu 16.04 LTS: $ sudo service cloudhsm-client stop
   - Windows: C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
2. **Update the AWS CloudHSM Configuration Files**
   - Amazon Linux, CentOS 6, RHEL 6, Amazon Linux 2, CentOS 7, RHEL 7, Ubuntu 16.04 LTS: $ sudo /opt/cloudhsm/bin/configure -a <HSM ENI IP>
   - Windows: C:\Program Files\Amazon\CloudHSM>configure.exe -a <HSM ENI IP>
3. **Start the AWS CloudHSM Client**
   - Amazon Linux, CentOS 6, RHEL 6: $ sudo start cloudhsm-client
   - Amazon Linux 2, CentOS 7, RHEL 7, Ubuntu 16.04 LTS: $ sudo service cloudhsm-client start
   - Windows client 1.1.2+: C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
   - Windows 1.1.1 and older: C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
4. **Update the cloudhsm_mgmt_util Configuration File**
   - Amazon Linux, CentOS 6, RHEL 6, Amazon Linux 2, CentOS 7, RHEL 7, Ubuntu 16.04 LTS: $ sudo /opt/cloudhsm/bin/configure -m
   - Windows: C:\Program Files\Amazon\CloudHSM>configure.exe -m

   For more information on cloudhsm_mgmt_util commands and examples of using the commands, see cloudhsm_mgmt_util Command Reference.

**Managing Keys**

The key_mgmt_util command line tool helps Crypto Users (CU) manage keys in the HSMs. It includes multiple commands that generate, delete, import, and export keys, get and set attributes, find keys, and perform cryptographic operations.

**Setting Up key_mgmt_util**

Complete the following setup before you use key_mgmt_util:

1. **Start the AWS CloudHSM Client**

   Before you use key_mgmt_util, you must start the AWS CloudHSM client.

2. **Start key_mgmt_util**

   - `Amazon Linux, CentOS 6, RHEL 6, Amazon Linux 2, CentOS 7, RHEL 7, Ubuntu 16.04 LTS: $ sudo /opt/cloudhsm/bin/ key_mgmt_util`
   - • `Windows: C:\Program Files\Amazon\CloudHSM>key_mgmt_util.exe`

   For more information about the commands, see key_mgmt_util Command Reference.

**Helper Tools**

The Configure Tool updates your CloudHSM client configuration files. This enables the AWS CloudHSM to synchronize the HSMs in a cluster.

These tools help you to use the tools and software libraries.

Verify the Performance of the HSM measures the performance of your HSM hardware independent of software libraries.

## Monitoring AWS CloudHSM Audit Logs

When an HSM in your account receives a command from the AWS CloudHSM command line tools or software libraries, it records its execution of the command in audit log form.

**Interpreting HSM Audit Logs**

The events in the HSM audit logs have standard fields. Some event types have additional fields that capture useful information about the event. For example, user login and user management events include the user name and user type of the user. Key management commands include the key handle.

An example of CloudHSM Audit logs looks like this:

```
Time: 12/19/17 21:01:17.174902, usecs:1513717277174902
Sequence No : 0x2
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_APPLIANCE_USER (0xfc)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : app_user
User Type : CN_APPLIANCE_USER (5)
```

**Auditing Log Reference**

AWS CloudHSM records HSM management commands in audit log events. Each event has an operation code (Opcode) value that identifies the action that occurred and its response. You can use the Opcode values to search, sort, and filter the logs. For more information, see Audit Log Reference.

# Installing the Connector

## Prerequisites

AWS Cloudwatch installer requires the following

- An Amazon VPC, with EC2 instance.
- A public subnet and a private subnet. For more information on how to create a public and private subnet, see Configuring an Amazon Virtual Private Cloud.
- Make sure that you are able to connect from an EC2 instance in a public subnet to an EC2 instance in private subnet.
- An IAM user with the required privileges. For more information on how to create the IAM user, see "Setting User Permissions in AWS Account " on the next page
- Create an S3 bucket, then create a folder to store fault tolerance files. To create a S3 bucket, see, AWS Documentation.
- Copy the CloudWatch installer to the EC2 instance in a private subnet by SSH through a public subnet.

## Configuring an Amazon Virtual Private Cloud and Subnets

To configure the existing Amazon VPC, you must create a private subnet and associate it with the lambda function.

A private subnet is a subnet with a route table pointing to a Nat gateway.

**To create a subnet:**

1. Create an internet gateway if you do not have one.
2. From the VPC console, go to the navigation pane and select **Subnets**.
3. Select an existing subnet, or to create a new subnet, select **Create Subnet**.
4. Select the **Route Table** tab, and then select **Edit**.
5. From the **Change to:** drop-down menu, select an appropriate route table.

   For a public subnet, the default route must point to an internet gateway.

**To create a NAT gateway:**

1. From the VPC console, go to the navigation panel, select **NAT Gateways**, and then select **Create NAT Gateway**.

2. In the **Subnet** field, select the public subnet already created.
3. In the **Elastic IP Allocation ID** field, select an existing Elastic IP address, or select **Create New EIP**, then select **Create a NAT Gateway**.

**To create a route table:**

1. In the VPC console, select **Route Tables**, then select **Create Route Table**.
2. Specify a name, select the **VPC** drop-down menu and select your VPC, then select **Yes, Create**.
3. Select the new route table, and then select the **Routes tab**.
4. Select **Edit**, and then select **Add another route**.

   **Destination:** 0.0.0.0/0

   **Target:** Private subnet with the NAT gateway created.

## Setting User Permissions in AWS Account

To install the AWS Connector you require an AWS account, for more information, see AWS Documentation. You can either use your main account with your AWS CloudWatch Connector or, create a new user from your main AWS account, and give this user privileges to deploy and work with the AWS CloudWatch Connector.

**To set user permissions:**

1. Create a new policy.
2. From the AWS Dashboard, select **AWS Cloudwatch Connector**.
3. Select **Policies**.
4. Select **Create Policy**.
5. Go to the **JSON** tab.

   From the JSON editor, paste the following JSON document:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": [
"cloudformation:DescribeStacks",
"cloudformation:UpdateStack",
"cloudformation:CreateStack",
```

```
"cloudformation:DeleteStack",
"cloudformation:DescribeStackResources"],
"Resource":[
"arn:aws:cloudformation:*:*:stack/*/*"
]
},
{
"Sid":"VisualEditor1",
"Effect":"Allow",
"Action":[
"cloudformation:ListStacks"],
"Resource": "*"
}
]
}
```

6. Select **Review Policy**.
7. Save the Policy.
   a. On the field **Name**, enter CloudFormationBasicExecution.
   b. Select **Create Policy**. A message is displayed of the policy is successfully created.
8. Create a policy from the AWS Dashboard:
   a. Select **Policies**.
   b. Select **Create Policy**
   c. Go to the **JSON** tab.

      From the JSON editor, paste the following JSON document:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": [
"lambda:CreateFunction",
"lambda:AddPermission",
"lambda:DeleteFunction",
"lambda:UpdateFunctionCode",
"lambda:DeleteEventSourceMapping",
```

```
"lambda:InvokeAsync",
"lambda:UpdateFunctionConfiguration",
"lambda:UpdateEventSourceMapping",
"lambda:RemovePermission",
"lambda:InvokeFunction",
"lambda:CreateEventSourceMapping",
"iam:PutRolePolicy",
"iam:AddRoleToInstanceProfile",
"iam:PutGroupPolicy",
"iam:DeleteRole",
"iam:PassRole",
"iam:DeleteRolePolicy",
"iam:CreatePolicy",
"iam:AttachGroupPolicy",
"iam:PutUserPolicy",
"iam:DeleteGroup",
"iam:CreateRole",
"iam:AttachRolePolicy",
"iam:CreateGroup",
"iam:AttachUserPolicy",
"iam:AddUserToGroup",
"logs:DeleteSubscriptionFilter",
"logs:PutSubscriptionFilter",
"logs:DeleteLogStream",
"logs:CreateLogStream",
"logs:CreateLogGroup",
"logs:DeleteLogGroup",
"logs:PutLogEvents",
"logs:PutRetentionPolicy",
"kinesis:PutRecords",
"kinesis:RemoveTagsFromStream",
"kinesis:AddTagsToStream",
"kinesis:CreateStream",
"kinesis:PutRecord",
"kinesis:DeleteStream",
"ec2:DeleteRouteTable",
"ec2:CreateNetworkInterfacePermission",
```

```
"ec2:CreateRoute",
"ec2:RevokeSecurityGroupEgress",
"ec2:CreateTags",
"ec2:DisassociateRouteTable",
"ec2:CreateNetworkInterface",
"ec2:DescribeSecurityGroups",
"ec2:DeleteFlowLogs",
"ec2:AssociateRouteTable",
"ec2:CreateTrafficMirrorFilterRule",
"ec2:DeleteNetworkInterface",
"ec2:CreateRouteTable",
"ec2:CreateFlowLogs",
"ec2:CreateTrafficMirrorFilter",
"ec2:DeleteTags",
"ec2:DescribeNetworkInterfaces",
"ec2:CreateSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:DetachNetworkInterface",
"ec2:DeleteRoute",
"ec2:DeleteSecurityGroup",
"events:RemoveTargets",
"events:PutRule",
"events:DeleteRule",
"events:PutTargets",
"s3:DeleteObject",
"s3:PutObject"
],
"Resource": "*"
}
]
}
```

d. Select Review Policy.

e. Save the Policy.

f. On the field name, enter **LimitedPermission**.

g. Select **Create Policy**. A message is displayed of the policy is successfully created.

h. Create a user.

   i. On the AWS Dashboard, select **IAM**.
   ii. Select **Users**.
   iii. Select **Add User**.
i. Specify a name for the new user.
j. Go to **Access Type** and select **Programmatic Access** and **AWS Management Console Access**.
k. Select your Console password.
   i. Select **Auto Generated Password** and AWS creates a random password.
   ii. Select **Custom password** and select your custom password.
   iii. Users can change the password on the first time login. For more information, see, Require Password Reset.
l. Select **Next**.

   You are taken to **Permissions**.
m. Select **Attach existing policies directly**.
n. Click the checkboxes of the following policies:
   i. CloudFormationBasicExecution created above, and the following AWS managed policies
   ii. LimitedPermission created above, and the following AWS managed policies
   iii. AmazonEC2ReadOnlyAccess
   iv. CloudWatchEventsReadOnlyAccess
   v. AmazonKinesisReadOnlyAccess
   vi. AmazonS3ReadOnlyAccess
   vii. CloudWatchLogsReadOnlyAccess
   viii. IAMReadOnlyAccess
   ix. AWSLambdaReadOnlyAccess
o. Go to **Tags** > **Review** > **Create User**.

   A message is displayed, indicating the user creation was successful.
p. Note the **Access Key ID** and the **Secret Access Key**. They are required for deployment.

# Installing Syslog NG Daemon as a Forwarding Agent

Because AWS CloudWatch is a Cloud-native Connector, you must configure Syslog NG Daemon smartconnector as a forwarding agent to collect CloudWatch events.

1. Launch the EC2 Instance in a public subnet and a private subnet.
2. Launch a terminal emulator, the log in to the public EC2 instance using the key.
3. Upload the key of the private EC2 instance to the public EC2 instance.
4. From the public EC2 instance, run the following command:

   `chmod 600 testprivate.pem`
5. SSH to the private instance using the following command:

   `ssh ec2-user@private-ip-address -itestprivate.pem`
6. Using an appropriate terminal emulator, upload the Syslog NG Daemon installer to public EC2 instance.
7. Use the following command to copy the Syslog NG Daemon installer to the private EC2 instance:

   `scp -i testprivate.pem ArcSight-<versionnumber>.0-Connector-Linux64.bin ec2-user@private-ip-address:/home/ec2-user/.`
8. Install and configure the Syslog NG Daemon SmartConnector in the private instance.
9. Select 1.0 as the CEF File version.
10. Configure the protocol as default TLS.
11. Configure the port.
12. Select CSV File or CEF File as the destination. If you use any ArcSight product such as Logger or ESM, select the destination appropriately.

    **Note**: To emit the Avro output, select Transformation Hub as the destination.
13. Run the SmartConnector as a standalone process or as a service.
14. Upload the `<ARCSIGHT_HOME>/current/user/agent/remote_management.p12` certificate to the */certs* folder in the S3 bucket.

# Configuring the Load Balancer as a Destination

In environments where the event load is more than what can be handled by a single Syslog NG Daemon SmartConnector, you can configure Load Balancer to handle large event loads. For more information about configuring Load Balancer, see Configuring Load Balancer.

After you install and configure Load Balancer, copy the *<install_folder>*/current/user/loadbalancer/loadbalancer.p12 certificate file to the */certs* folder in the S3 bucket.

# Opening Ports

You must ensure that the ports on the server on which you installed the Syslog NG Daemon SmartConnector is accessible from AWS. To open ports, use one of the following procedures depending on whether you have installed Syslog NG Daemon SmartConnector on a virtual machine or not.

### Opening Ports on a Non-Virtual Machine

If you installed Syslog NG Daemon SmartConnector on a physical, non-virtual machine, ensure that the ports on which you installed it are accessible to AWS.

### Opening Ports on a Virtual Machine

If you have installed the Syslog NG Daemon SmartConnector on a virtual machine in AWS, ensure that the ports on which you installed Syslog NG Daemon SmartConnector are open in both AWS and the virtual machine.

**To open inbound ports on AWS:**

**Note**: These steps are applicable only for Windows instances and not Linux EC2 instances.

1. Log in to AWS as a user with administrator privileges.
2. Go to **Services** and select **EC2**.
3. Select **Instances**.
4. Choose the EC2 instance to be edited.
5. Click **Launch-Wizard**.
6. Edit the **Inboud** and **Outbound** fields as required.

**To open ports in the virtual server:**

1. Log in to the virtual AWS machine.
2. Open the AWS Firewall.
3. Click Inbound **Rules** > **New Rule** > **Port** > **Next** > **TCP** > **Specific local ports**
4. Enter the same port or port range on which you installed the Syslog NG Daemon SmartConnector.
5. Click **Next** > **Allow the connection** > **Next** > **Profile** > **Next**.
6. Name the rule.
7. Click **Finish**.

# Deploying the Connector

This section provides information about deploying the AWS CloudWatch Connector to collect and forward events from AWS CloudWatch to an ArcSight Syslog NG Daemon SmartConnector or an ArcSight Load Balancer, then send the events an ArcSight destination.

## Configuring AWS Credentials

You must specify the AWS credentials if you are executing the installer.sh file for the first time. If not, you can update the AWS credentials when prompted.

1. Provide execution rights to all the .sh files.
2. Execute the installer.sh file and enter your AWS Access Key ID.
3. Enter your AWS Secret Access Key.
4. Select the region.

   You are taken to the main screen.

## Deploying the Connector

1. From the main menu, select **Deploy**.

   S3 buckets are scanned for analysis.
2. Specify a name for the stack. AWS resources created during deployment will be located here.

   **Note**: Stack names must be unique on each region and they must have a valid format.
3. Specify whether to create Flow Log Components or use an existing log group.
   a. To create a new flow log group, select **Yes**. The installer creates a flow log group under the selected VPC. Specify a retention value in days from the following permitted values:

      1,3,5,7,14,30,60,90,120,150,180,365,400,545,731,1827,3653.

      The retention days are applied to newly created flow logs group under the VPC.
   b. To use an existing log group, select **No**. Log Groups must exist in the current Region and they cannot be previously subscribed to another service.
4. Enter the **Log Groups** names. Every log group is monitored and then sent to the event processing Lambda.
5. Select the VPC from which the AWS CloudWatch connector must gather the events.
6. Select the S3 bucket in which the event processing jar and properties file are located.
7. Enter the path for the jar key.

The format is: folderName/subFolderName/jarKeyName.jar

**Note**: If an invalid format is entered, an error message is displayed.

8. Specify the external properties file.
   The format is: folderName/subFolderName/external.properties
9. Select the private subnet created previously.
10. Enter the port number specified in the properties file.
11. Click **Yes** to confirm deployment summary.
    If the deployment is successful, a message is displayed.

# Updating the Connector

You can update the connector parameters such as the Stack, selected VPC, specified S3 bucket, the path for the jar key and jar key name, the path and the properties file name, or the Subnet and the port number specified during connector deployment.

**To update the connector:**

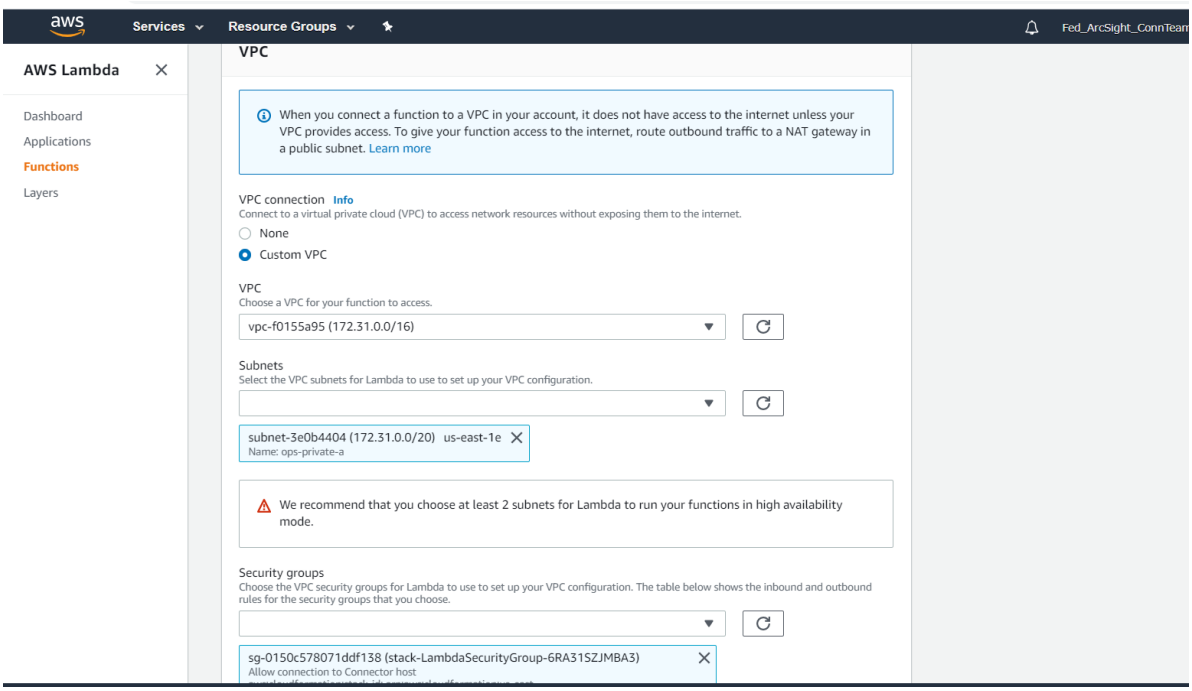
1. From the main menu, select **Update**.
   S3 buckets are scanned for analysis.
2. Select the Stack you want to update.
3. Create or update the specified Log Groups.
4. Select a VPC to update.
5. Select an S3 bucket to update.
6. Update the jar key path or name.
7. Update external properties key path or name.
8. Select a **Subnet** to update.
9. Update the port number.
10. Click **Yes** to confirm your changes and proceed with the udpate.
    A message is displayed, If the update was successful.

# Post-Deployment Configurations

## Enabling Lambda VPC

To enable a Lambda VPC:

1. From VPC Connection, select Custom VPC.
2. Select the VPC and subnet created while following the "Prerequisites" on page 17 steps.
3. From **Security Grops**, select the **Lambda Security Group** created during connector deployment and click **save**.




## Running Lambda Functions in High Availability

When the subnet runs out of IP addresses or if the availability zone goes down, AWS displays a warning.

You can run functions in high availability mode.

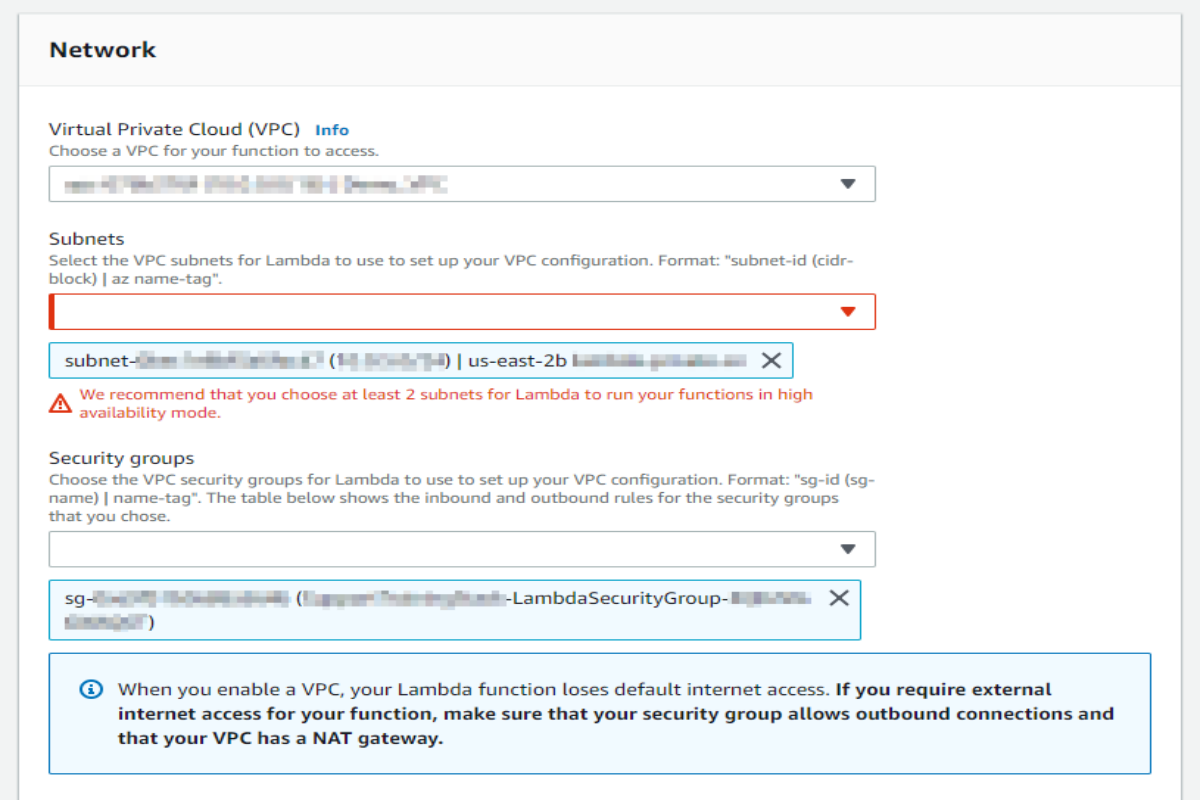
1. From **Services** > **Network** > **Lambda** > **Functions**.
2. Select the Lambda Function created.
3. Click **Network**.

For more information, see AWS Documentation .

## Removing Lambda's Subnet Warning

If the subnet is only used for Lambda events, open a browser. It is very unlikely for the subnet to run out of IP addresses as it is only used by CloudwatchConnectorEventProcessing and Hearthbeat Lambda Functions in the stack.

If an availability zone goes down and cannot communicate with the Connector, the events are not sent to the destination, and instead, they are stored in a S3 bucket. The moment, the availability zones comes back online, the Lambda Event Processing function processes these events first, before processing any new events.




**To add multiple subnets to Lambda functions, manually:**

1. Create a new private Subnet in your VPC.

   The destination of the new private subnet cannot be located in the same availability zone in which your original subnet was created.
2. Identify the Lambda functions from your stack STACKNAME-CloudwatchConnectorEventProcessing and STACKNAME-Heartbeat.
3. From each Lambda function, go to **Network** and select your recently created subnet from the dropdown.
4. Click **Save**, the warning should not be displayed.

# Managing the Connector

## Overriding Internal Parser Files with Custom Parser Files

You can override the internal parsers with the custom parsers. The parser format looks like this:

```
# CEF fields

# any alphanumeric value will be included in the CEF result

# any value that starts with @ and followed by a number 0-13

# will be replaced with the corresponding flow log field value in the CEF result

# HEADER fields, all fields start with Upper case

DeviceVendor=Arcsight

DeviceProduct=Cloudwatch Connector

DeviceVersion=@0

DeviceEventClassID=CW

Name=AWS Flow Log Event

Severity=0

# EXTENSION fields, all fields in lower case

duid=@1

deviceInboudInterface=@2

src=@3

dst=@4

spt=@5

dpt=@6

proto=@7

cn=@8

in=@9

start=@10

end=@11

act=@12
```

```
cs=@13
```

After the parser files are created, save them as VPC Flow Logs, Route 53 Logs, or CloudHSM Audit Logs. Custom parsers and internal parser files are the same; however, the mapping fields change.

| Parser Category | File name |
|---|---|
| VPC Flow logs | vpc.map |
| Route53 logs | route53.map |
| CloudHSM Audit Logs | cloudhsm.map |

**Note:** If a parser file does not have the correct name, it cannot be recognized.

**To override an internal parser file**:

1. Create a custom parser file and upload it to an S3 bucket and a directory that is dedicated to parser overrides.

   The S3 bucket and the directory are defined in the **external.properties** file. For more information, see "Prerequisites" on page 17.

2. Override an existing parser file in the Lambda function.

   It updates the Event Processor Lambda Function.

   To override a file and restart the connector, do one of the following:

   - Use the AWS command-line interface (CLI):

     ```
     aws lambda update-function-code --function-name <value> --s3-bucket <value> --s3-key <jar file path>
     ```

     For more information, see update-function-code.

   - Use the AWS Console:

     a. Go to the jar file path that is configured in the S3 bucket.

     b. Click the **Object URL** icon and copy the URL.

     c. Go to the Lambda Console.

     d. In the **Function Code** group box, click **Action**, then select **Upload a file from Amazon S3**.

     e. In the **Amazon S3 link URL** box, paste the object URL you copied, then click **Save**.

# Upgrading the Connector

The upgrade allows to do a binary upgrade of the AWS CloudWatch connector. A binary upgrade, enables you to continue using the components created during deployment. Your custom settings should not be affected.

It's necessary to update the path or the name of the jar key that contains the binary changes.

**To upgrade the connector:**

1. Run the installer.sh .
2. Select the **Stack** you are upgrading.

   The script returns the data with the preferences entered during installation or a previous update.
3. A confirmation screen pops, click **Yes**.

   The "Enter a Jar Key" screen is displayed.
4. Update the path and/or the name of the jar key and confirm your changes. You are taken to the Update Status Screen. When completed, a message is displayed, confirming the update was successful.

   **Note:** You may update other values. if necessary.

# Updating Parser Files

ArcSight SmartConnector for AWS CloudWatch is designed to support parser file updates at run time without any code changes.

The updates extend events of supported services. You can apply parser updates on the basic installation of connectors.

You can apply the following parser updates:

- Updates received from OpenText. See "Updating Parser Files" above.
- Updates received from a support team or other sources. See "Updating Parser Files" above.

## Apply Monthly Parser Updates from OpenText

The monthly updates for ArcSight SmartConnector parser releases are available at the ArcSight Marketplace. To set up your administrative account and download the parser updates, refer to ArcSight Marketplace at: https://marketplace.microfocus.com/arcsight

The updates for all parsers are available from OpenText as an **ArcSight-24.2xxxx.0-ConnectorParsers.aup** package.

**To apply the parser updates**:

1. Download the **ArcSight-24.2xxxx.0-ConnectorParsers.aup** package from the ArcSight Marketplace.
2. To apply monthly parser updates to Cloud Connectors:
   a. Download the **`ArcSight-24.2xxxx.0-aup-extractor.jar`** utility from the location where you have downloaded the connector.

      **Note**: Your system must have Java 1.8.x or later version installed and Java available in the operating system's path to use the aup-extractor.jar utility

   b. Specify the following command to use the utility to extract parser files from the package:

      `java -jar aup-extractor.jar <AUP filename>`

      Examples:

      - `java -jar aup-extractor.jar ArcSight-24.2xxxx.0-ConnectorParsers.aup` - When the **.aup** package is in the same directory where the JAR file is present.
      - `java -jar aup-extractor.jar c:\MyFolder\ArcSight-24.2xxxx.0-ConnectorParsers.aup` - When the **.aup** package is present in other directory.

      You can either provide one or both the parameters. If you do not provide any parameters, the utility picks up any available. aup file and creates a new folder named **output** in the directory from where the utility is run and uploads the output files.

      The following folders will be extracted:

      - **aws_cloudwatch**: Contains security parser for AWS Cloudwatch.
      - **aws_securityhub**: Contains security parser for AWS Security Hub.
      - **azure_emitter**: Contains security parser for Azure emitter.

   c. Copy the parser files in the **output/aws_cloudwatch** folder and upload them to the AWS environment. To upload:

      i. Navigate to your AWS S3 bucket. For more information, see "Prerequisites" on page 17.

      ii. Search for the **Maps** folder and upload the new parser files.

   d. Follow the step and sub-steps in Override an existing parser file in the Lambda function.

## Apply the Parser Updates from a Support Team or Other Sources

In this case, the updated parser files are already available, which you need to add in the AWS environment.

**To add updated parser files provided by a Support Team or other sources to the AWS environment**:

a. Navigate to your AWS S3 bucket. For more information, see "Prerequisites" on page 17.

b. Search for the **Maps** folder and upload the new parser files.

c. Follow the step and sub-steps in Override an existing parser file in the Lambda function

# Undeploying the CloudWatchConnector

**To undeploy the CloudWatch connector:**

1. From the main menu, select **Undeploy**.
2. Select the Stack to be undeployed.
3. Click **Yes**, to confirm undeployment.

   You are taken to the Deleting Network Interfaces screen. When the process completes successfully, a message is displayed.

# Appendix

## Troubleshooting

### The connector configuration on CentOS/RHEL AWS EC2 instances fails and displays the error "Connection refused" or "Unable to get the list of supported connectors for VM [Container1]"

The following error is displayed when you install the connector on an EC2 instance of CentOS or RHEL OS:

"`Connection refused`" or "`Unable to get the list of supported connectors for VM [Container1]`"

This error might occur because of low memory.

This error might also be displayed when installing the SyslogNGDaemon SmartConnector in EC2 instances that are created with 1GB or 2GB memory.

**Workaround:**

The Java heap memory has been increased to 1GB. Therefore, the EC2 instance must be 2GB.

To fix this issue, create the EC2 instance with +2GB and proceed with the connector installation

# Sample External Properties File

Follow the template to create an external properties file:

```
##
## External properties file to upload to s3
##
#
# Valid properties
#
#Arcsight connector host name or ip address, required parameter
host.name=0.0.0.0
#
#Arcsight connector port number, required parameter
port.number=9999
#
#Arsight connector certificate bucket name in s3, required parameter
certificate.bucket=bucket_name
#
#Arsight connector certificate key location in s3, required parameter
certificate.key=path/to/file
#
#Arsight connector certificate password, required parameter
certificate.password=password
#
#Number of retries if the destination does not respond,
#if de destination stills without responding
#the mechanism of transport.cache will be activated
send.retries=3
#
#Arcsight connector transport cache bucket name in s3, required parameter
transport.cache.bucket=bucket_name
```

```
#
#Arcsight connector transport cache location in s3, required parameter
transport.cache.directory=path/to/file
#
#Log Level changes the log level to the specified level
#value can be any of: info debug error all warn fatal "trace" "off" or
#case insensitive value
log.level=info debug error all warn fatal trace off
```

The port must be same used for the ArcSight Syslog NG Daemon SmartConnector or the ArcSight Load Balancer. This value is also required during deployment.

The path of the certificate and its password uploaded in step 3 must be added to the `[properties.file]`. The file should look like this:

```
host.name="10.0.0.0
port.number=1514
certificate.bucket=vlc-s3-bucket
certificate.key=cert/remote_management.p12
certificate.password=changeit
send.retries=3
transport.cache.bucket=vlc-s3-bucket
transport.cache.directory=transport.cache
log.level=info
```

# Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this computer, click the link above and an email window opens with the following information in the subject line:

**Feedback on Configuration Guide for Amazon Web Services Cloudwatch SmartConnector (SmartConnectors 24.2)**

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to MFI-Documentation-Feedback@opentext.com.

We appreciate your feedback!