aws

User Guide

Amazon Kinesis Agent for Microsoft Windows

Amazon Kinesis Agent for Microsoft Windows: User Guide

Table of Contents

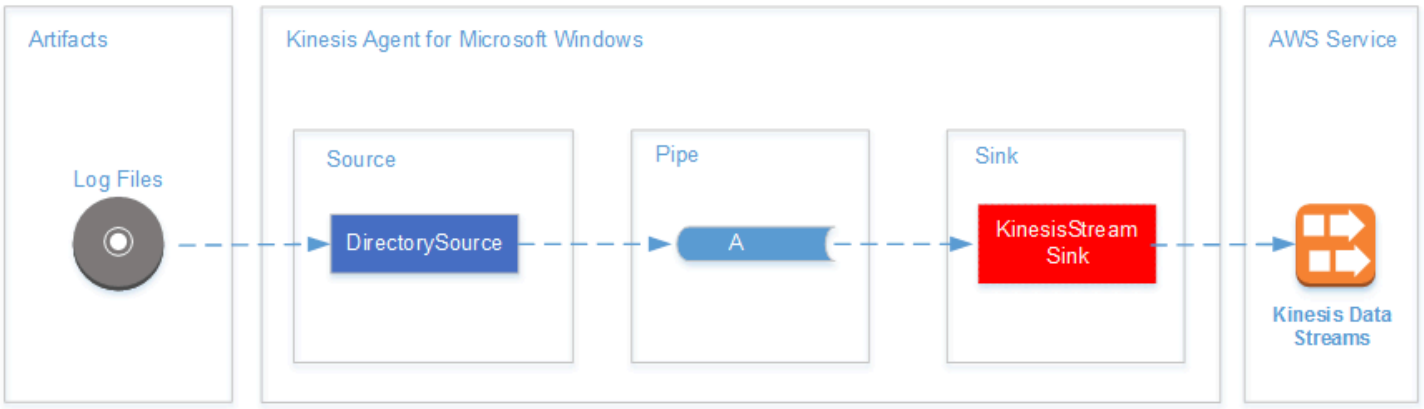
What Is Amazon Kinesis Agent for Microsoft Windows?

Amazon Kinesis Agent for Microsoft Windows (Kinesis Agent for Windows) is a configurable and extensible agent. It runs on fleets of Windows desktop computers and servers, either on-premises or in the AWS Cloud. Kinesis Agent for Windows efficiently and reliably gathers, parses, transforms, and streams logs, events, and metrics to various AWS services, including Kinesis Data Streams, Firehose, Amazon CloudWatch, and CloudWatch Logs.

From those services, you can then store, analyze, and visualize the data using a variety of other AWS services, including the following:

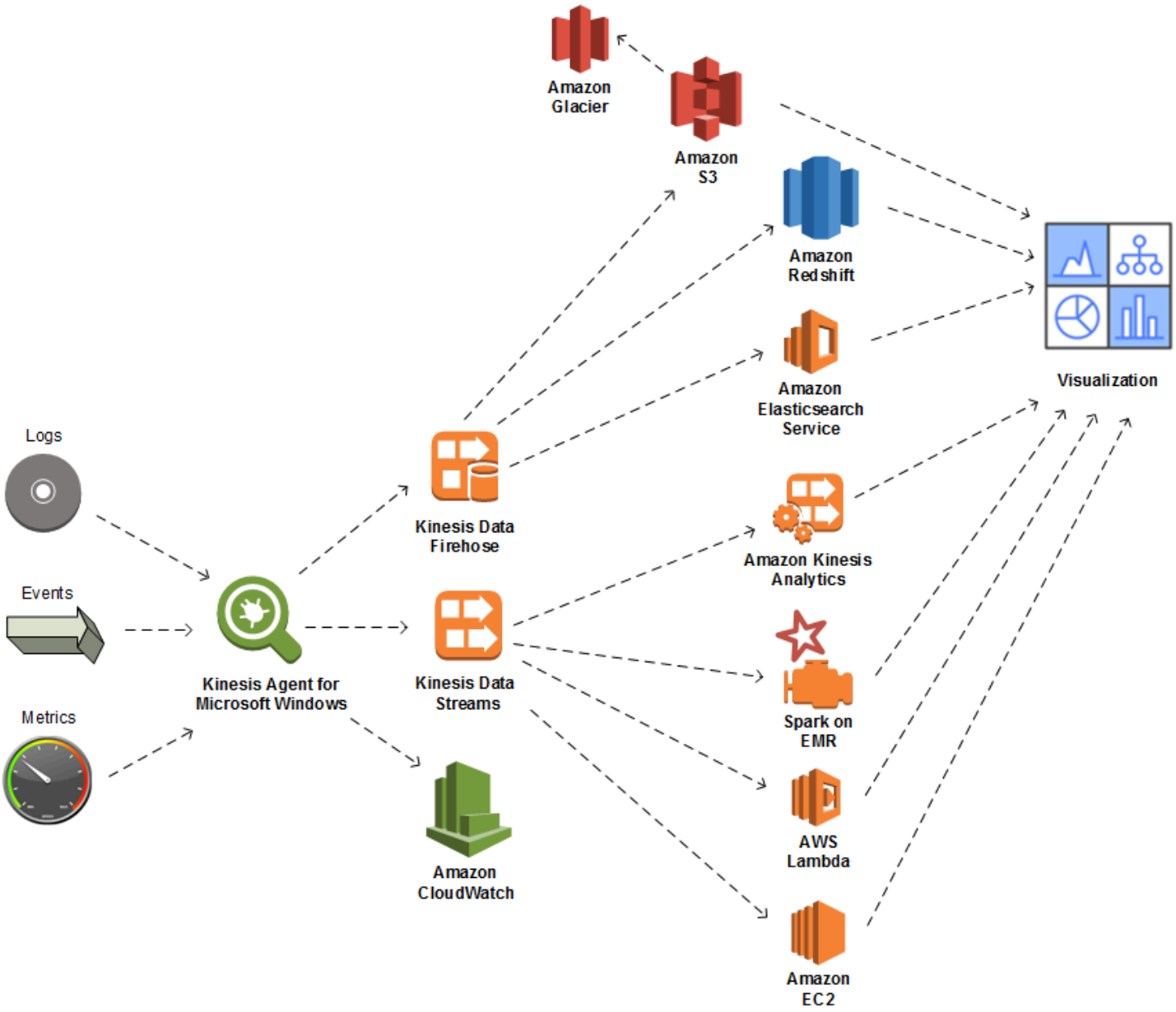
- Amazon Simple Storage Service (Amazon S3)
- Amazon Redshift
- Amazon OpenSearch Service (Amazon ES)
- Managed Service for Apache Flink
- Amazon Quick
- Amazon Athena
- Kibana

The following diagram illustrates a simple configuration of Kinesis Agent for Windows that streams log files to Kinesis Data Streams.



For more information about sources, pipes, and sinks, see Amazon Kinesis Agent for Microsoft Windows Concepts.

The following diagram illustrates some of the ways you can build custom, real-time data pipelines using stream-processing frameworks. These frameworks include Managed Service for Apache Flink, Apache Spark on Amazon EMR, and AWS Lambda.



Topics

- About AWS
- What Can You Do with Kinesis Agent for Windows?
- Benefits
- Getting Started with Kinesis Agent for Windows

About AWS

Amazon Web Services (AWS) is a collection of digital infrastructure services that you can use when developing applications. The services include computing, storage, database, analytics, and application synchronization (messaging and queuing). AWS uses a pay-as-you-go service model. You are charged only for the services that you—or your applications—use. Also, to make its services more approachable for prototyping and experimentation, AWS offers a free usage tier. On this tier, services are free below a certain level of usage. For more information about AWS costs and the *Free Tier*, see the Getting Started Resource Center. To create an AWS account, open the AWS home page and sign up.

What Can You Do with Kinesis Agent for Windows?

Kinesis Agent for Windows provides the following features and capabilities:

Collect Logs, Events, and Metrics Data

Kinesis Agent for Windows collects, parses, transforms, and streams logs, events, and metrics from fleets of servers and desktops to one or more AWS services. The payload received by the services can be in a different format from the original source. For example, a log might be stored in a particular textual format (such as syslog format) on a server. Kinesis Agent for Windows can collect and parse that text and optionally transform it to JSON format, for example, before streaming to AWS. This facilitates simpler processing by some AWS services that consume JSON. Data streamed to Kinesis Data Streams can be continuously processed by Managed Service for Apache Flink to generate additional metrics and aggregated metrics, which in turn can power live dashboards. You can store the data using a variety of AWS services (such as Amazon S3) depending on how the data is used downstream in a data pipeline.

Integrate with AWS services

You can configure Kinesis Agent for Windows to send log files, events, and metrics to several different AWS services:

- Firehose — Easily store streamed data in Amazon S3, Amazon Redshift, OpenSearch Service, or Splunk for further analysis.
- Kinesis Data Streams — Process streamed data using custom applications hosted in Managed Service for Apache Flink or Apache Spark on Amazon EMR. Or use custom code running on Amazon EC2 instances, or custom serverless functions running in AWS Lambda.
- CloudWatch — View streamed metrics in graphs, which you can combine into dashboards. Then set CloudWatch alarms that are triggered by metric values that breach preset thresholds.
- CloudWatch Logs — Store streamed logs and events, and view and search them in the AWS Management Console, or process them further downstream in a data pipeline.

Install and Configure Quickly

You can install and configure Kinesis Agent for Windows in just a few steps. For more information, see Installing Kinesis Agent for Windows and Configuring Amazon Kinesis Agent for Microsoft Windows. A simple declarative configuration file specifies the following:

- The sources and formats of logs, events, and metrics to gather.
- The transformations to apply to the gathered data. Additional data can be included, and existing data can be transformed and filtered.
- The destinations where the final data is streamed, and the buffering, sharding, and format for the streaming payloads.

Kinesis Agent for Windows comes with built-in parsers for log files generated by common Microsoft enterprise services such as:

- Microsoft Exchange
- SharePoint
- Active Directory domain controllers
- DHCP servers

No Ongoing Administration

Kinesis Agent for Windows automatically adapts to various situations without losing any data. These include log rotation, recovery after reboot, and temporary network or service interruptions. You can configure Kinesis Agent for Windows to automatically update to new versions. No operator intervention is required in any of these situations.

Extend Using Open Architecture

If the declarative capabilities and built-in plugins are insufficient for monitoring server or desktop systems, you can extend Kinesis Agent for Windows by creating plugins. New plugins enable new sources and destinations for logs, events, and metrics. The source code for Kinesis Agent for Windows is available at Kinesis Agent windows.

Benefits

Kinesis Agent for Windows performs the initial data gathering, transformation, and streaming for logs, events, and metrics for data pipelines. Building these data pipelines has numerous benefits:

Analysis and Visualization

The integration of Kinesis Agent for Windows with Firehose and its transformation capabilities make it easy to integrate with several different analytic and visualization services:

- Quick — A cloud-based BI service that can ingest from many different sources. Kinesis Agent for Windows can transform data and stream it to Amazon S3 and Amazon Redshift via Firehose. This process enables discovery of deep insights from the data using Quick visualizations.

- Athena — An interactive query service that enables SQL-based querying of data. Kinesis Agent for Windows can transform and stream data to Amazon S3 via Firehose. Athena can then interactively execute SQL queries against that data to rapidly inspect and analyze logs and events.
- Kibana — An open-source data visualization tool. Kinesis Agent for Windows can transform and stream data to OpenSearch Service via Firehose. You can then use Kibana to explore that data. Create and open different visualizations, including histograms, line graphs, pie charts, heat maps, and geospatial graphics.

Security

A log and event data analysis pipeline that includes Kinesis Agent for Windows can detect and alert on security breaches in organizations, which can help you block or stop attacks.

Application Performance

Kinesis Agent for Windows can collect logs, events, and metric data about application or service performance. A complete data pipeline can then analyze this data. This analysis helps you improve your application and service performance and reliability by detecting and reporting on defects that otherwise might not be apparent. For example, you can detect significant changes in the execution times of service API calls. When correlated to a deployment, this capability helps you locate and resolve new performance problems with services that you own.

Service Operations

A data pipeline can analyze the data collected to predict potential operational issues and provide insight into how to avoid service outages. For example, you can analyze logs, events, and metrics to determine current and projected capacity usage so that you can bring additional capacity online before service users are affected. If a service outage occurs, you can analyze the data to determine the impact on customers during the outage period.

Auditing

A data pipeline can process the logs, events, and metrics that Kinesis Agent for Windows collects and transforms. You can then audit this processed data using various AWS services. For example, Firehose could receive a data stream from Kinesis Agent for Windows, which stores the data in Amazon S3. You could then audit this data by executing interactive SQL queries using Athena.

Archiving

Often the most important operational data is data that is recently collected. However, analysis of data that is collected about applications and services over several years can also be useful, for example, for long range planning. Keeping large amounts of data can be expensive. Kinesis Agent for Windows can collect, transform, and store data in Amazon S3 via Firehose. Therefore, Amazon Glacier is available to reduce the costs of archiving older data.

Alerting

Kinesis Agent for Windows streams metrics to CloudWatch. In turn, you can create CloudWatch alarms to send a notification via Amazon Simple Notification Service (Amazon SNS) when a metric consistently violates a specific threshold. This gives engineers a better awareness of the operational issues with their applications and services.

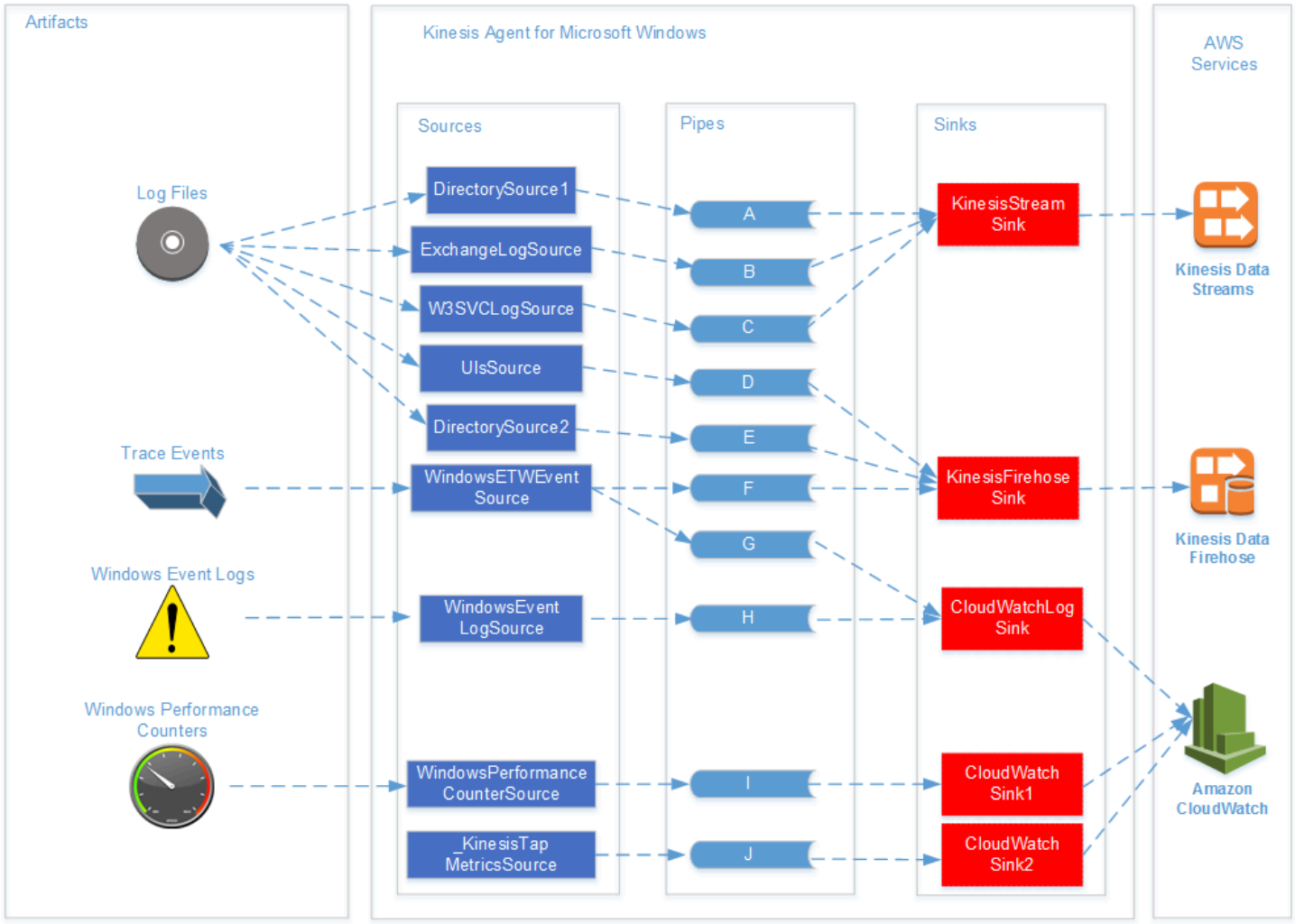
Getting Started with Kinesis Agent for Windows

To learn more about Kinesis Agent for Windows, we recommend that you start with the following sections:

- Amazon Kinesis Agent for Microsoft Windows Concepts
- Getting Started with Amazon Kinesis Agent for Microsoft Windows

Amazon Kinesis Agent for Microsoft Windows Concepts

Understanding the key concepts of Amazon Kinesis Agent for Microsoft Windows (Kinesis Agent for Windows) can make it easier for you to collect and stream data on desktop and server fleets to the remainder of the data pipeline for processing.



This diagram of a data pipeline illustrates the following components and processes:

Servers and desktops have artifacts like log files, events, and metrics that are gathered by one or more Kinesis Agent for Windows *sources*. The data can be optionally transformed from, for example, a flat file text format to an object.

The data (in either object or text form) can then flow into one or more Kinesis Agent for Windows *pipes*. A pipe connects one source to one Kinesis Agent for Windows *sink*. The pipe can optionally filter out unnecessary data.

A sink can optionally transform data parsed into objects into JSON or XML. The sink sends the data to a specific AWS service, such as Kinesis Data Streams, Firehose, or Amazon CloudWatch.

Using multiple pipes, a single source can send the same data to multiple sinks (for example, see pipes **F** and **G** in the diagram). Using multiple pipes, different sources can stream data to a single sink (for example, see pipes **A**, **B**, and **C** in the diagram). It is also possible to use multiple pipes to stream data from multiple sinks to multiple sources. Sources, sinks, and pipes have types, and there can be more than one source, sink, or pipe of the same type.

For examples of configuration files that declare sources, sinks, and pipes, see Kinesis Agent for Windows Configuration Examples.

Topics

Data Pipelines

A *data pipeline* is used to gather, process, visualize, and possibly generate alarms for applications and services. Kinesis Agent for Windows fits into data pipelines at the start—where logs, events, and metrics are gathered from fleets of desktop computers or servers. Kinesis Agent for Windows streams the collected data to the various AWS services that form the rest of the data pipeline. A data pipeline has a purpose, such as visualizing the health of a particular service in real time to help engineers more effectively operate that service. A service health data pipeline can do any of the following:

- Alert engineers to problems before those problems affect the experience for customers of the services.
- Help engineers efficiently manage the cost of the service by showing resource usage trends. These trends enable them to adjust resource levels appropriately, or even implement automatic scaling scenarios.
- Provide insight into the root cause of problems that are reported by customers of the service. This speeds up the resolution of those problems and reduces support costs.

For a step-by-step example of constructing a data pipeline using Kinesis Agent for Windows, see Tutorial: Stream JSON Log Files to Amazon S3 Using Kinesis Agent for Windows.

Sources

A Kinesis Agent for Windows *source* gathers logs, events, or metrics. A source gathers a particular kind of data from a particular producer of that data based on the type of the source. For example, the `DirectorySource` type gathers log files from particular directories in the file system. If the data isn't already structured (as with some kinds of log files), a source can be useful in parsing the textual representation into some structured form. Each source corresponds to a particular *source declaration* in the Kinesis Agent for Windows `appsettings.json` configuration file. The source declaration provides essential details for configuring the source to tailor the source based on the specific data gathering requirements. The kinds of details that can be configured vary by source type. For example, the `DirectorySource` source type requires specification of the directory where the log files reside.

For more details about source types and source declarations, see Source Declarations.

Sinks

A Kinesis Agent for Windows *sink* takes data gathered by a Kinesis Agent for Windows source and streams that data to one of several possible AWS services that form the rest of the data pipeline. Each sink corresponds to a particular *sink declaration* in the Kinesis Agent for Windows `appsettings.json` configuration file. The sink declaration provides essential details for configuring the sink to tailor the sink based on the specific data streaming requirements. The kinds of details that can be configured vary by sink type. For example, some sink types allow a sink declaration to specify a particular serialization `Format` for the data provided to them. When this option is specified in the sink declaration, serialization of the gathered data occurs before streaming the data to the AWS service that is associated with the sink.

For more information about sink types and sink declarations, see Sink Declarations.

Pipes

A Kinesis Agent for Windows *pipe* connects the output of a Kinesis Agent for Windows source to the input of a Kinesis Agent for Windows sink. It optionally transforms the data as it flows through the pipe. Each pipe corresponds to a particular pipe declaration in the Kinesis Agent for

Windows `appsettings.json` configuration file. The pipe declaration provides essential details for configuring the sink, such as the source and sink for the pipe.

For more information about pipe types and pipe declarations, see Pipe Declarations.

Getting Started with Amazon Kinesis Agent for Microsoft Windows

You can use Amazon Kinesis Agent for Microsoft Windows (Kinesis Agent for Windows) to collect, parse, transform, and stream logs, events, and metrics from your Windows fleet to various AWS services. The following information contains prerequisites and step-by-step instructions for installing and configuring Kinesis Agent for Windows.

Topics

Prerequisites

Before installing Kinesis Agent for Windows, ensure that you have the following prerequisites:

- Familiarity with Kinesis Agent for Windows concepts. For more information, see Amazon Kinesis Agent for Microsoft Windows Concepts.
- An AWS account for using the various AWS services related to your data pipeline. For information about creating and configuring an AWS account, see Setting Up an AWS account.
- Microsoft .NET Framework 4.6 or later on each desktop or server that will run Kinesis Agent for Windows. For more information, see Install the .NET Framework for developers in the Microsoft .NET documentation.

  To determine the latest version of the .NET Framework that is installed on a desktop or server, use the following PowerShell script:

```
[System.Version](
(Get-ChildItem 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP' -recurse `
| Get-ItemProperty -Name Version -ErrorAction SilentlyContinue `
| Where-Object { ($_.PSChildName -match 'Full') } `
| Select-Object Version | Sort-Object -Property Version -Descending)[0]).Version
```

- The streams where you want to send data from Kinesis Agent for Windows (if using Amazon Kinesis Data Streams). Create the streams using the Kinesis Data Streams console, the AWS CLI, or AWS Tools for Windows PowerShell. For more information, see Creating and Updating Data Streams in the *Amazon Kinesis Data Streams Developer Guide*.
- The Firehose delivery streams where you want to send data from Kinesis Agent for Windows (if using Amazon Data Firehose). Create delivery streams using the Firehose console, the AWS CLI, or AWS Tools for Windows PowerShell. For more information, see Creating an Amazon Data Firehose Delivery Stream in the *Amazon Data Firehose Developer Guide*.

Setting Up an AWS account

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

To sign up for an AWS account

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

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

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

To create an administrator user, choose one of the following options.

Choose one way to manage your administrator	To	By	You can also
In IAM Identity Center (Recommended)	Use short-term credentials to access AWS. This aligns with the security best practices. For information about best practices, see Security best practices in IAM in the *IAM User Guide*.	Following the instructions in Getting started in the *AWS IAM Identity Center User Guide*.	Configure programmatic access by Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*.
In IAM (Not recommended)	Use long-term credentials to access AWS.	Following the instructions in Create an IAM user for emergency access in the *IAM User Guide*.	Configure programmatic access by Manage access keys for IAM users in the *IAM User Guide*.

To sign up for AWS and create an administrator account

1. If you don't have an AWS account, open https://aws.amazon.com/. Choose **Create an AWS account**, and then follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.
2. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
3. In the navigation pane, choose **Groups**, and then choose **Create New Group**.

4. For **Group Name**, enter a name for your group, such as **`Administrators`**, and then choose **Next Step**.
5. In the list of policies, select the check box next to the **AdministratorAccess** policy. You can use the **Filter** menu and the **Search** box to filter the list of policies.
6. Choose **Next Step**. Choose **Create Group**, and your new group appears under **Group Name**.
7. In the navigation pane, choose **Users**, and then choose **Create New Users**.
8. In box **1**, enter a user name, clear the check box next to **Generate an access key for each user**, and then choose **Create**.
9. In the list of users, choose the name (not the check box) of the user that you just created. You can use the **Search** box to search for the user name.
10. Choose the **Groups** tab, and then choose **Add User to Groups**.
11. Select the check box next to the administrators group, and then choose **Add to Groups**.
12. Choose the **Security Credentials** tab. Under **Sign-In Credentials**, choose **Manage Password**.
13. Select **Assign a custom password**, enter a password in the **Password** and **Confirm Password** boxes, and then choose **Apply**.

Installing Kinesis Agent for Windows

There are three ways that you can install Kinesis Agent for Windows on Windows:

- Install using MSI (a Windows installer package).
- Install from AWS Systems Manager, a set of services for administering servers and desktops.
- Run a PowerShell script.

> **Note**
>
> The following instructions occasionally use the terms *KinesisTap* and *`AWSKinesisTap`*. These words mean the same thing as Kinesis Agent for Windows, but you must specify them as-is when executing these instructions.

Install Kinesis Agent for Windows using MSI

You can download the latest Kinesis Agent for Windows MSI package from the kinesis-agent-windows repository on GitHub. After you download the MSI, use Windows to launch it and follow the installer prompts. After installation, you can uninstall as you would any Windows application.

Alternatively, you can use the msiexec command from the Windows command prompt to install silently, turn on logging, and uninstall as shown in the following examples. Replace *AWSKinesisTap.1.1.216.4.msi* with the appropriate version of Kinesis Agent for Windows for your application.

To install Kinesis Agent for Windows silently:

```
msiexec /i AWSKinesisTap.1.1.216.4.msi /q
```

To log installation messages for troubleshooting in a file named *logfile.log*:

```
msiexec /i AWSKinesisTap.1.1.216.4.msi /q /L*V logfile.log
```

To uninstall Kinesis Agent for Windows using the command prompt:

```
msiexec.exe /x {ADAB3982-68AA-4B45-AE09-7B9C03F3EBD3} /q
```

Install Kinesis Agent for Windows using AWS Systems Manager

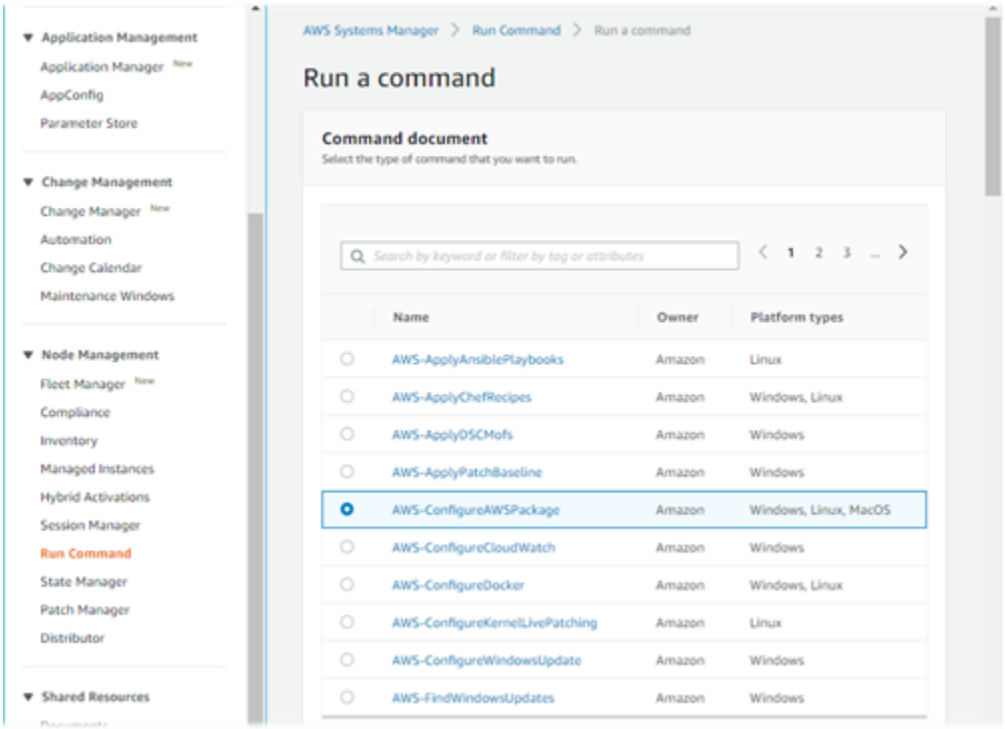
Follow these steps to install Kinesis Agent for Windows using Systems Manager Run Command. For more information about Run Command, see AWS Systems Manager Run Command in the *AWS Systems Manager User Guide*. In addition to using Systems Manager Run Command, you can also use Systems Manager Maintenance Windows and State Manager to automate the deployment of Kinesis Agent for Windows over time.

> **Note**
>
> Systems Manager installation for Kinesis Agent for Windows is available in the AWS Regions listed in AWS Systems Manager except the following:
>
> - cn-north-1
> - cn-northwest-1

- All AWS GovCloud Regions.

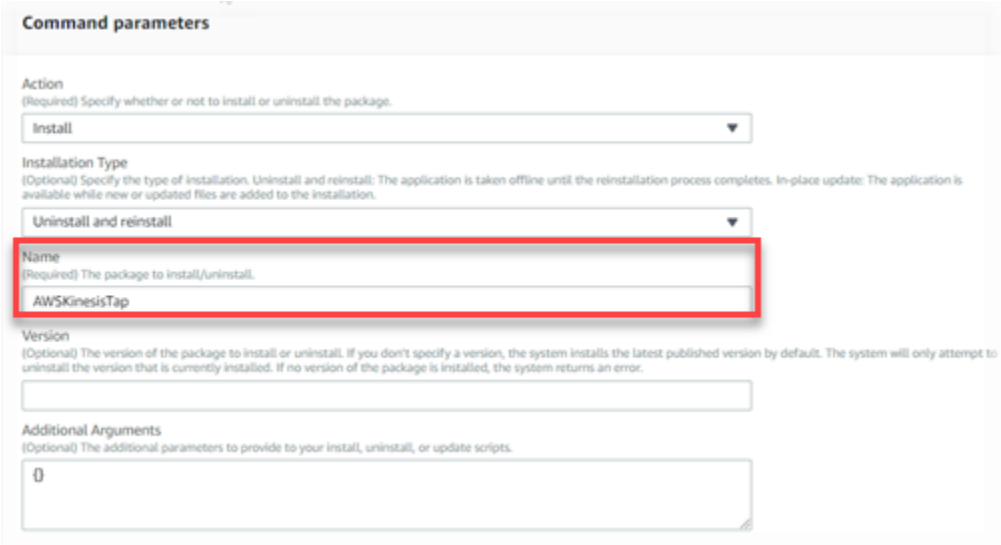
To install Kinesis Agent for Windows using Systems Manager

1. Ensure that version 2.2.58.0 or later of the SSM Agent is installed on instances where you want to install Kinesis Agent for Windows. For more information, see Installing and configuring SSM Agent on Windows instances in the *AWS Systems Manager User Guide*.
2. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
3. From the navigation pane, under **Node Management**, choose **Run Command**, and then choose **Run Command**.
4. From the **Command document** list, select the **`AWS-ConfigureAWSPackage`** document.



5. Under **Command Parameters**, for **Name**, enter **`AWSKinesisTap`**. Leave other settings to their defaults.

> **Note**
>
> Leave **Version** blank to specify the latest version of the `AWSKinesisTap` package. Optionally, you can enter a specific version to install.



6. Under **Targets**, specify the instances on which to run the command. You can choose to specify instances based on tags associated with instances, you can choose instances manually, or you can specify a resource group that includes instances.
7. Leave all other settings to their defaults and choose **Run**.

Install Kinesis Agent for Windows Using PowerShell

Use a text editor to copy the following commands into a file and save it as a PowerShell script. We use `InstallKinesisAgent.ps1` in the following example.

```
Param(
    [ValidateSet("prod", "beta", "test")]
    [string] $environment = 'prod',
    [string] $version,
    [string] $baseurl
)

# Self-elevate the script if required.
if (-Not ([Security.Principal.WindowsPrincipal]
 [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltI
 'Administrator')) {
    if ([int](Get-CimInstance -Class Win32_OperatingSystem | Select-Object -
ExpandProperty BuildNumber) -ge 6000) {
        $CommandLine = '-File "' + $MyInvocation.MyCommand.Path + '" ' +
 $MyInvocation.UnboundArguments
```

```
        Start-Process -FilePath PowerShell.exe -Verb Runas -ArgumentList $CommandLine
        Exit
    }
}

# Allows input to change base url. Useful for testing.
if ($baseurl) {
    if (!$baseUrl.EndsWith("/")) {
        throw "Invalid baseurl param value. Must end with a trailing forward slash
 ('/')"
    }

    $kinesistapBaseUrl = $baseurl
} else {
    $kinesistapBaseUrl = "https://s3-us-west-2.amazonaws.com/kinesis-agent-windows/
downloads/"
}

Write-Host "Using $kinesistapBaseUrl as base url"

$webClient = New-Object System.Net.WebClient

try {
    $packageJson = $webClient.DownloadString($kinesistapBaseUrl + 'packages.json' + '?
_t=' + [System.DateTime]::Now.Ticks) | ConvertFrom-Json
} catch {
    throw "Downloading package list failed."
}



if ($version) {
    $kinesistapPackage = $packageJson.packages | Where-Object { $_.packageName -eq
 "AWSKinesisTap.$version.nupkg" }

    if ($null -eq $kinesistapPackage) {
        throw "No package found matching input version $version"
    }
} else {
    $packageJson = $packageJson.packages | Where-Object { $_.packageName -match
 ".nupkg" }
    $kinesistapPackage = $packageJson[0]
}

$packageName = $kinesistapPackage.packageName
```

```
$checksum = $kinesistapPackage.checksum

#Create %TEMP%/kinesistap if not exists
$kinesistapTempDir = Join-Path $env:TEMP 'kinesistap'
if (![System.IO.Directory]::Exists($kinesistapTempDir)) {[void]
[System.IO.Directory]::CreateDirectory($kinesistapTempDir)}

#Download KinesisTap.x.x.x.x.nupkg package
$kinesistapNupkgPath = Join-Path $kinesistapTempDir $packageName
$webClient.DownloadFile($kinesistapBaseUrl + $packageName, $kinesistapNupkgPath)
$kinesistapUnzipPath = $kinesistapNupkgPath.Replace('.nupkg', '')

# Calculates hash of downloaded file. Downlevel compatible using .Net hashing on PS < 4
if ($PSVersionTable.PSVersion.Major -ge 4) {
    $calculatedHash = Get-FileHash $kinesistapNupkgPath -Algorithm SHA256
    $hashAsString = $calculatedHash.Hash.ToLower()
} else {
    $sha256 = New-Object System.Security.Cryptography.SHA256CryptoServiceProvider
    $calculatedHash =
 [System.BitConverter]::ToString($sha256.ComputeHash([System.IO.File]::ReadAllBytes($kinesistap
    $hashAsString = $calculatedHash.Replace("-", "").ToLower()
}

if ($checksum -eq $hashAsString) {
 Write-Host 'Local file hash matches checksum.' -ForegroundColor Green
} else {
 throw ("Get-FileHash does not match! Package may be corrupted.")
}

#Delete Unzip path if not empty
if ([System.IO.Directory]::Exists($kinesistapUnzipPath)) {Remove-Item –Path
 $kinesistapUnzipPath -Recurse -Force}

#Unzip KinesisTap.x.x.x.x.nupkg package
$null =
 [System.Reflection.Assembly]::LoadWithPartialName('System.IO.Compression.FileSystem')
[System.IO.Compression.ZipFile]::ExtractToDirectory($kinesistapNupkgPath,
 $kinesistapUnzipPath)

#Execute chocolaeyInstall.ps1 in the package and wait for completion.
$installScript = Join-Path $kinesistapUnzipPath '\tools\chocolateyInstall.ps1'
& $installScript

# Verify service installed.
```

```
$serviceName = 'AWSKinesisTap'
$service = Get-Service -Name $serviceName -ErrorAction Ignore
if ($null -eq $service) {
    throw ("Service not installed correctly.")
} else {
    Write-Host "Kinesis Tap Installed." -ForegroundColor Green
    Write-Host "After configuring run the following to start the service: Start-Service
 -Name $serviceName." -ForegroundColor Green
}
```

Open an elevated command prompt window. In the directory where the file was downloaded, use the following command to run the script:

```
PowerShell.exe -File ".\InstallKinesisAgent.ps1"
```

To install a specific version of Kinesis Agent for Windows, add the `-version` option:

```
PowerShell.exe -File ".\InstallKinesisAgent.ps1" -version "version"
```

Replace *`version`* with a valid Kinesis Agent for Windows version number. For version information, see the kinesis-agent-windows repository on GitHub.

There are many deployment tools which can remotely execute PowerShell scripts. They can be used to automate the installation of Kinesis Agent for Windows on fleets of servers or desktops.

Configuring and Starting Kinesis Agent for Windows

After installing Kinesis Agent for Windows, you must configure and start the agent. After that, no further operation intervention should be required.

To configure and start Kinesis Agent for Windows

1. Create and deploy a Kinesis Agent for Windows configuration file. This file configures sources, sinks, and pipes, along with other global configuration items.

   For more information about Kinesis Agent for Windows configuration, see Configuring Amazon Kinesis Agent for Microsoft Windows.

   For complete configuration file examples that you can customize and install, see Kinesis Agent for Windows Configuration Examples.

2. Open an elevated PowerShell command prompt window, and start Kinesis Agent for Windows using the following PowerShell command:

```
Start-Service -Name AWSKinesisTap
```

Configuring Amazon Kinesis Agent for Microsoft Windows

Before starting Amazon Kinesis Agent for Microsoft Windows, you must create a configuration file and deploy it. The configuration file provides the necessary information to collect, transform, and stream data on Windows servers and desktop computers to various AWS services. Configuration files define sets of sources, sinks, and pipes that connect sources to sinks, along with optional transformations.

The Kinesis Agent for Windows configuration file is named `appsettings.json`. Deploy this file to `%PROGRAMFILES%\Amazon\AWSKinesisTap`.

Topics

Basic Configuration Structure

The basic structure of the Amazon Kinesis Agent for Microsoft Windows configuration file is a JSON document with the following template:

```
{
     "Sources": [ ],
     "Sinks": [ ],
     "Pipes": [ ]
}
```

- The value of `Sources` is one or more Source Declarations.
- The value of `Sinks` is one or more Sink Declarations.

- The value of `Pipes` is one or more Pipe Declarations.

For more information about the Kinesis Agent for Windows source, pipe, and sink concepts, see Amazon Kinesis Agent for Microsoft Windows Concepts.

The following example is a complete `appsettings.json` configuration file that configures Kinesis Agent for Windows to stream Windows application log events to Firehose.

```
{
  "Sources": [
    {
      "LogName": "Application",
      "Id": "ApplicationLog",
      "SourceType": "WindowsEventLogSource"
    }
  ],
  "Sinks": [
    {
      "StreamName": "ApplicationLogFirehoseStream",
      "Region": "us-west-2",
      "Id": "MyKinesisFirehoseSink",
      "SinkType": "KinesisFirehose"
    }
  ],
  "Pipes": [
    {
      "Id": "ApplicationLogTotestKinesisFirehoseSink",
      "SourceRef": "ApplicationLog",
      "SinkRef": "MyKinesisFirehoseSink"
    }
  ]
}
```

For information about each kind of declaration, see the following sections:

- Source Declarations
- Sink Declarations
- Pipe Declarations

Configuration Case Sensitivity

JSON-formatted files are typically case sensitive, and you should assume that all the keys and values in Kinesis Agent for Windows configuration files are also case sensitive. Some keys and values in the `appsettings.json` configuration file are not case sensitive; for example:

- The value of the `Format` key-value pair for sinks. For more information, see Sink Declarations.
- The value of the `SourceType` key-value pair for sources, the `SinkType` key-value pair for sinks, and the `Type` key-value pair for pipes and plugins.
- The value of `RecordParser` key-value pair for the `DirectorySource` source. For more information, see DirectorySource Configuration.
- The value of the `InitialPosition` key-value pair for sources. For more information, see Bookmark Configuration.
- Prefixes for variable substitutions. For more information, see Configuring Sink Variable Substitutions.

Source Declarations

In Amazon Kinesis Agent for Microsoft Windows, *source declarations* describe where and what log, event, and metric data should be collected. They also optionally specify information for parsing that data so that it can be transformed. The following sections describe configurations for the built-in source types that are available in Kinesis Agent for Windows. Because Kinesis Agent for Windows is extensible, you can add custom source types. Each source type typically requires specific key-value pairs in the configuration objects that are relevant for that source type.

All source declarations must contain at least the following key-value pairs:

`Id`

A unique string that identifies a particular source object within the configuration file.

`SourceType`

The name of the source type for this source object. The source type specifies the origin of the log, event, or metric data that is being collected by this source object. It also controls what other aspects of the source can be declared.

For examples of complete configuration files that use different kinds of source declarations, see Streaming from Various Sources to Kinesis Data Streams.

Topics

DirectorySource Configuration

Overview

The `DirectorySource` source type gathers logs from files that are stored in the specified directory. Because log files come in many different formats, the `DirectorySource` declaration lets you specify the format of the data in the log file. Then you can transform the log contents to a standard format such as JSON or XML before streaming to various AWS services.

The following is an example `DirectorySource` declaration:

```
{
    "Id": "myLog",
    "SourceType": "DirectorySource",
    "Directory": "C:\\Program Data\\MyCompany\\MyService\\logs",
    "FileNameFilter": "*.log",
    "IncludeSubdirectories": true,
    "IncludeDirectoryFilter": "cpu\\cpu-1;cpu\\cpu-2;load;memory",
    "RecordParser": "Timestamp",
    "TimestampFormat": "yyyy-MM-dd HH:mm:ss.ffff",
```

```
    "Pattern": "\\d{4}-\\d{2}-\\d(2}",
    "ExtractionPattern": "",
    "TimeZoneKind": "UTC",
    "SkipLines": 0,
    "Encoding": "utf-16",
    "ExtractionRegexOptions": "Multiline"
}
```

All `DirectorySource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string `"DirectorySource"` (required).

`Directory`

The path to the directory containing the log files (required).

`FileNameFilter`

Optionally limits the set of files in the directory where log data is collected based on a wild card file-naming pattern. If you have multiple log file name patterns, this feature allows you to use a single `DirectorySource`, as shown in the following example.

```
FileNameFilter: "*.log|*.txt"
```

System administrators sometimes compress log files before archiving them. If you specify `"*.*"` in `FileNameFilter`, known compressed files are now excluded. This feature prevents `.zip`, `.gz`, and `.bz2` files from being streamed accidentally. If this key-value pair is not specified, data from all files in the directory are collected by default.

`IncludeSubdirectories`

Specifies to monitor subdirectories to arbitrary depth limited by the operating system. This feature is useful for monitoring web servers with multiple websites. You can also use the `IncludeDirectoryFilter` attribute to monitor only certain subdirectories specified in the filter.

`RecordParser`

Specifies how the `DirectorySource` source type should parse the log files that are found in the specified directory. This key-value pair is required, and the valid values are as follows:

- `SingleLine` — Each line of the log file is a log record.
- `SingleLineJson` — Each line of the log file is a JSON-formatted log record. This parser is useful when you want to add additional key-value pairs to the JSON using object decoration. For more information, see Configuring Sink Decorations. For an example that uses the `SingleLineJson` record parser, see Tutorial: Stream JSON Log Files to Amazon S3 Using Kinesis Agent for Windows.
- `Timestamp` — One or more lines can include a log record. The log record starts with a timestamp. This option requires specifying the `TimestampFormat` key-value pair.
- `Regex` — Each record starts with text that matches a particular regular expression. This option requires specifying the `Pattern` key-value pair.
- `SysLog` — Indicates that the log file is written in the syslog standard format. The log file is parsed into records based on that specification.
- `Delimited` — A simpler version of the Regex record parser where data items in the log records are separated by a consistent delimiter. This option is easier to use and executes faster than the Regex parser, and it is preferred when this option is available. When using this option, you must specify the `Delimiter` key-value pair.

`TimestampField`

Specifies which JSON field contains the timestamp for the record. This is only used with the `SingleLineJson` `RecordParser`. This key-value pair is optional. If it is not specified, Kinesis Agent for Windows uses the time when the record was read for the timestamp. One advantage of specifying this key-value pair is that latency statistics generated by Kinesis Agent for Windows are more accurate.

`TimestampFormat`

Specifies how to parse the date and time associated with the record. The value is either the string `epoch` or a .NET date/time format string. If the value is `epoch`, time is parsed based on UNIX Epoch time. For more information about UNIX Epoch time, see Unix time. For more information about .NET date/time format strings, see Custom Date and Time Format Strings in the Microsoft .NET documentation). This key-value pair is required only if the `Timestamp` record parser is specified, or the `SingleLineJson` record parser is specified along with the `TimestampField` key-value pair.

`Pattern`

Specifies a regular expression that must match the first line of a potentially multi-line record. This key-value pair is only required for the `Regex` record parser.

`ExtractionPattern`

Specifies a regular expression that should use named groups. The record is parsed using this regular expression and the named groups form the fields of the parsed record. These fields are then used as the basis for constructing JSON or XML objects or documents that are then streamed by sinks to various AWS services. This key-value pair is optional, and is available with the `Regex` record parser and the Timestamp parser.

The `Timestamp` group name is specially processed, as it indicates to the `Regex` parser which field contains the date and time for each record in each log file.

`Delimiter`

Specifies the character or string that separates each item in each log record. This key-value pair must be (and can only be) used with the `Delimited` record parser. Use the two-character sequence `\t` to represent the tab character.

`HeaderPattern`

Specifies a regular expression for matching the line in the log file that contains the set of headers for the record. If the log file does not contain any header information, use the `Headers` key-value pair to specify the implicit headers. The `HeaderPattern` key-value pair is optional and only valid for the `Delimited` record parser.

> **Note**
>
> An empty (0 length) header entry for a column causes the data for that column to be filtered from the final output of the `DirectorySource` parsed output.

`Headers`

Specifies the names for the columns of data parsed using the specified delimiter. This key-value pair is optional and only valid for the `Delimited` record parser.

> **Note**
>
> An empty (0 length) header entry for a column causes the data for that column to be filtered from the final output of the `DirectorySource` parsed output.

`RecordPattern`

Specifies a regular expression that identifies lines in the log file that contain record data. Other than the optional header line identified by `HeaderPattern`, lines that do not match the specified `RecordPattern` are ignored during record processing. This key-value pair is optional and only valid for the `Delimited` record parser. If it is not provided, the default is to consider any line that does not match the optional `HeaderPattern` or the optional `CommentPattern` to be a line that contains parseable record data.

`CommentPattern`

Specifies a regular expression that identifies lines in the log file that should be excluded before parsing the data in the log file. This key-value pair is optional and only valid for the `Delimited` record parser. If it is not provided, the default is to consider any line that does not match the optional `HeaderPattern` to be a line that contains parseable record data, unless `RecordPattern` is specified.

`TimeZoneKind`

Specifies whether the timestamp in the log file should be considered in the local time zone or the UTC time zone. This is optional and defaults to UTC. The only valid values for this key-value pair are `Local` or UTC. The timestamp is never altered if `TimeZoneKind` is either not specified or if the value is UTC. The timestamp is converted to UTC when the `TimeZoneKind` value is `Local` and the sink receiving the timestamp is CloudWatch Logs, or the parsed record is sent to other sinks. Dates and times that are embedded in messages are not converted.

`SkipLines`

When specified, controls the number of lines ignored at the start of each log file before record parsing occurs. This is optional, and the default value is 0.

Encoding

By default, Kinesis Agent for Windows can automatically detect the encoding from bytemark. However, the automatic encoding may not work correctly on some older unicode formats. The following example specifies the encoding required to stream a Microsoft SQL Server log.

```
"Encoding": "utf-16"
```

For a list of encoding names, see List of encodings in Microsoft .NET documentation.

ExtractionRegexOptions

You can use `ExtractionRegexOptions` to simplify regular expressions. This key-value pair is optional. The default is `"None"`.

The following example specifies that the `"."` expression matches any character including `\r\n`.

```
"ExtractionRegexOptions" = "Multiline"
```

For a list of the possible fields for ExtractionRegexOptions, see the RegExOptions Enum in Microsoft .NET documentation.

Regex Record Parser

You can parse unstructured text logs using the `Regex` record parser along with the `TimestampFormat`, `Pattern`, and `ExtractionPattern` key-value pairs. For example, suppose that your log file looks like the following:

```
[FATAL][2017/05/03 21:31:00.534][0x00003ca8][0000059c][][ActivationSubSystem]
[GetActivationForSystemID][0] 'ActivationException.File: EQCASLicensingSubSystem.cpp'
[FATAL][2017/05/03 21:31:00.535][0x00003ca8][0000059c][][ActivationSubSystem]
[GetActivationForSystemID][0] 'ActivationException.Line: 3999'
```

You can specify the following regular expression for the `Pattern` key-value pair to help break the log file into individual log records:

```
^\[\w+\]\[(?<TimeStamp>\d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2}\.\d{3})\]
```

This regular expression matches the following sequence:

1. The start of the string being evaluated.
2. One or more word characters surrounded by square brackets.
3. A timestamp surrounded by square brackets. The timestamp matches the following sequence:
   a. A four-digit year

b. A forward slash
c. A two-digit month
d. A forward slash
e. A two-digit day
f. A space character
g. A two-digit hour
h. A colon
i. A two-digit minute
j. A colon
k. A two-digit second
l. A period
m. A three-digit millisecond

You can specify the following format for the `TimestampFormat` key-value pair to convert the textual timestamp into a date and time:

```
yyyy/MM/dd HH:mm:ss.fff
```

You can use the following regular expression for extracting the fields of the log record via the `ExtractionPattern` key-value pair.

```
^\[(?<Severity>\w+)\]\[(?<TimeStamp>\d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2}\.
\d{3})\]\[[^]]*\]\[[^]]*\]\[[^]]*\]\[(?<SubSystem>\w+)\]\[(?<Module>\w+)\]\[[^]]*\] '(?
<Message>.*)'$
```

This regular expression matches the following groups in sequence:

1. `Severity` — One or more word characters surrounded by square brackets.
2. `TimeStamp` — See the previous description for the timestamp.
3. Three unnamed square bracketed sequences of zero or more characters are skipped.
4. `SubSystem` — One or more word characters surrounded by square brackets.
5. `Module` — One or more word characters surrounded by square brackets.
6. One unnamed square bracketed sequence of zero or more characters is skipped.
7. One unnamed space is skipped.

8. `Message` — Zero or more characters surrounded by single quotes.

The following source declaration combines these regular expressions and the date time format to provide the complete instructions to Kinesis Agent for Windows for parsing this kind of log file.

```
{
    "Id": "PrintLog",
    "SourceType": "DirectorySource",
    "Directory": "C:\\temp\\PrintLogTest",
    "FileNameFilter": "*.log",
    "RecordParser": "Regex",
    "TimestampFormat": "yyyy/MM/dd HH:mm:ss.fff",
    "Pattern": "^\\[\\w+\\]\\[(?<TimeStamp>\\d{4}/\\d{2}/\\d{2} \\d{2}:\\d{2}:\\d{2}\\.
\\d{3})\\]",
    "ExtractionPattern": "^\\[(?<Severity>\\w+)\\]\\[(?<TimeStamp>\\d{4}/\\d{2}/\\d{2}
 \\d{2}:\\d{2}:\\d{2}\\.\\d{3})\\]\\[[^]]*\\]\\[[^]]*\\]\\[[^]]*\\]\\[(?<SubSystem>\\w
+)\\]\\[(?<Module>\\w+)\\]\\[[^]]*\\] '(?<Message>.*)'$",
    "TimeZoneKind": "UTC"
}
```

> **Note**
>
> Backslashes in JSON-formatted files must be escaped with an additional backslash.

For more information about regular expressions, see Regular Expression Language - Quick Reference in the Microsoft .NET documentation.

`Delimited` Record Parser

You can use the `Delimited` record parser to parse semistructured log and data files where there is a consistent character sequence separating each column of data in each row of data. For example, CSV files use a comma to separate each column of data, and TSV files use a tab.

Suppose that you want to parse a Microsoft NPS Database Format log file produced by a Network policy server. Such a file might look like the following:

```
"NPS-
MASTER","IAS",03/22/2018,23:07:55,1,"user1","Domain1\user1",,,,,,,,0,"192.168.86.137","Nate
 - Test 1",,,,,,,1,,0,"311 1 192.168.0.213 03/15/2018 08:14:29
 1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"Use Windows authentication for all users",1,,,,
```

```
"NPS-
MASTER","IAS",03/22/2018,23:07:55,3,,"Domain1\user1",,,,,,,,0,"192.168.86.137","Nate
 - Test 1",,,,,,1,,16,"311 1 192.168.0.213 03/15/2018 08:14:29
 1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"Use Windows authentication for all users",1,,,,
```

The following example `appsettings.json` configuration file includes a `DirectorySource` declaration that uses the `Delimited` record parser to parse this text into an object representation. It then streams JSON-formatted data to Firehose:

```
{
    "Sources": [
        {
            "Id": "NPS",
            "SourceType": "DirectorySource",
            "Directory": "C:\\temp\\NPS",
            "FileNameFilter": "*.log",
            "RecordParser": "Delimited",
            "Delimiter": ",",
            "Headers": "ComputerName,ServiceName,Record-Date,Record-Time,Packet-
Type,User-Name,Fully-Qualified-Distinguished-Name,Called-Station-ID,Calling-Station-
ID,Callback-Number,Framed-IP-Address,NAS-Identifier,NAS-IP-Address,NAS-Port,Client-
Vendor,Client-IP-Address,Client-Friendly-Name,Event-Timestamp,Port-Limit,NAS-Port-
Type,Connect-Info,Framed-Protocol,Service-Type,Authentication-Type,Policy-Name,Reason-
Code,Class,Session-Timeout,Idle-Timeout,Termination-Action,EAP-Friendly-Name,Acct-
Status-Type,Acct-Delay-Time,Acct-Input-Octets,Acct-Output-Octets,Acct-Session-Id,Acct-
Authentic,Acct-Session-Time,Acct-Input-Packets,Acct-Output-Packets,Acct-Terminate-
Cause,Acct-Multi-Ssn-ID,Acct-Link-Count,Acct-Interim-Interval,Tunnel-Type,Tunnel-
Medium-Type,Tunnel-Client-Endpt,Tunnel-Server-Endpt,Acct-Tunnel-Conn,Tunnel-Pvt-
Group-ID,Tunnel-Assignment-ID,Tunnel-Preference,MS-Acct-Auth-Type,MS-Acct-EAP-Type,MS-
RAS-Version,MS-RAS-Vendor,MS-CHAP-Error,MS-CHAP-Domain,MS-MPPE-Encryption-Types,MS-
MPPE-Encryption-Policy,Proxy-Policy-Name,Provider-Type,Provider-Name,Remote-Server-
Address,MS-RAS-Client-Name,MS-RAS-Client-Version",
            "TimestampField": "{Record-Date} {Record-Time}",
            "TimestampFormat": "MM/dd/yyyy HH:mm:ss"
        }
    ],
    "Sinks": [
        {
            "Id": "npslogtest",
            "SinkType": "KinesisFirehose",
            "Region": "us-west-2",
            "StreamName": "npslogtest",
            "Format": "json"
```

```
        }
    ],
    "Pipes": [
        {
            "Id": "W3SVCLog1ToKinesisStream",
            "SourceRef": "NPS",
            "SinkRef": "npslogtest"
        }
    ]
}
```

JSON-formatted data streamed to Firehose looks like the following:

```
{
    "ComputerName": "NPS-MASTER",
    "ServiceName": "IAS",
    "Record-Date": "03/22/2018",
    "Record-Time": "23:07:55",
    "Packet-Type": "1",
    "User-Name": "user1",
    "Fully-Qualified-Distinguished-Name": "Domain1\\user1",
    "Called-Station-ID": "",
    "Calling-Station-ID": "",
    "Callback-Number": "",
    "Framed-IP-Address": "",
    "NAS-Identifier": "",
    "NAS-IP-Address": "",
    "NAS-Port": "",
    "Client-Vendor": "0",
    "Client-IP-Address": "192.168.86.137",
    "Client-Friendly-Name": "Nate - Test 1",
    "Event-Timestamp": "",
    "Port-Limit": "",
    "NAS-Port-Type": "",
    "Connect-Info": "",
    "Framed-Protocol": "",
    "Service-Type": "",
    "Authentication-Type": "1",
    "Policy-Name": "",
    "Reason-Code": "0",
    "Class": "311 1 192.168.0.213 03/15/2018 08:14:29 1",
    "Session-Timeout": "",
    "Idle-Timeout": "",
```

```
    "Termination-Action": "",
    "EAP-Friendly-Name": "",
    "Acct-Status-Type": "",
    "Acct-Delay-Time": "",
    "Acct-Input-Octets": "",
    "Acct-Output-Octets": "",
    "Acct-Session-Id": "",
    "Acct-Authentic": "",
    "Acct-Session-Time": "",
    "Acct-Input-Packets": "",
    "Acct-Output-Packets": "",
    "Acct-Terminate-Cause": "",
    "Acct-Multi-Ssn-ID": "",
    "Acct-Link-Count": "",
    "Acct-Interim-Interval": "",
    "Tunnel-Type": "",
    "Tunnel-Medium-Type": "",
    "Tunnel-Client-Endpt": "",
    "Tunnel-Server-Endpt": "",
    "Acct-Tunnel-Conn": "",
    "Tunnel-Pvt-Group-ID": "",
    "Tunnel-Assignment-ID": "",
    "Tunnel-Preference": "",
    "MS-Acct-Auth-Type": "",
    "MS-Acct-EAP-Type": "",
    "MS-RAS-Version": "",
    "MS-RAS-Vendor": "",
    "MS-CHAP-Error": "",
    "MS-CHAP-Domain": "",
    "MS-MPPE-Encryption-Types": "",
    "MS-MPPE-Encryption-Policy": "",
    "Proxy-Policy-Name": "Use Windows authentication for all users",
    "Provider-Type": "1",
    "Provider-Name": "",
    "Remote-Server-Address": "",
    "MS-RAS-Client-Name": "",
    "MS-RAS-Client-Version": ""
}
```

SysLog Record Parser

For the SysLog record parser, the parsed output from the source includes the following information:

Attribute	Type	Description
SysLogTimeStamp	String	The original date and time from the syslog-formatted log file.
Hostname	String	The name of computer where the syslog-formatted log file resides.
Program	String	The name of the application or service that generated the log file.
Message	String	The log message generated by the application or service.
TimeStamp	String	The parsed date and time in ISO 8601 format.

The following is an example of SysLog data transformed into JSON:

```
{
    "SysLogTimeStamp": "Jun 18 01:34:56",
    "Hostname": "myhost1.example.mydomain.com",
    "Program": "mymailservice:",
    "Message": "Info: ICID 123456789 close",
    "TimeStamp": "2017-06-18T01:34.56.000"
}
```

Summary

The following is a summary of the key-value pairs available for the `DirectorySource` source and the `RecordParsers` related to those key-value pairs.

Key Name	RecordParser	Notes
SourceType	Required for all	Must have the value `Directory Source`

Key Name	RecordParser	Notes
Directory	Required for all	
FileNameFilter	Optional for all	
RecordParser	Required for all	
TimestampField	Optional for SingleLineJson	
TimestampFormat	Required for Timestamp, and required for SingleLineJson if TimestampField is specified	
Pattern	Required for Regex	
ExtractionPattern	Optional for Regex	Required for Regex if sink specifies json or xml format
Delimiter	Required for Delimited	
HeaderPattern	Optional for Delimited	
Headers	Optional for Delimited	
RecordPattern	Optional for Delimited	
CommentPattern	Optional for Delimited	
TimeZoneKind	Optional for Regex, Timestamp, SysLog, and SingleLineJson when a timestamp field is identified	

Key Name	RecordParser	Notes
SkipLines	Optional for all	

ExchangeLogSource Configuration

The ExchangeLogSource type is used to collect logs from Microsoft Exchange. Exchange produces logs in several different kinds of log formats. This source type parses all of them. Although it is possible to parse them using the DirectorySource type with the Regex record parser, it is much simpler to use the ExchangeLogSource. This is because you don't need to design and provide regular expressions for the log file formats. The following is an example ExchangeLogSource declaration:

```
{
   "Id": "MyExchangeLog",
   "SourceType": "ExchangeLogSource",
   "Directory": "C:\\temp\\ExchangeLogTest",
   "FileNameFilter": "*.log"
}
```

All exchange declarations can provide the following key-value pairs:

SourceType

Must be the literal string "ExchangeLogSource" (required).

Directory

The path to the directory containing the log files (required).

FileNameFilter

Optionally limits the set of files in the directory where log data is collected based on a wildcard file-naming pattern. If this key-value pair is not specified, then by default, log data from all files in the directory is collected.

TimestampField

The name of the column containing the date and time for the record. This key-value pair is optional and need not be specified if the field name is date-time or DateTime. Otherwise, it is required.

W3SVCLogSource Configuration

The `W3SVCLogSource` type is used to collect logs from Internet Information Services (IIS) for Windows.

The following is an example `W3SVCLogSource` declaration:

```
{
   "Id": "MyW3SVCLog",
   "SourceType": "W3SVCLogSource",
   "Directory": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
   "FileNameFilter": "*.log"
}
```

All `W3SVCLogSource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string `"W3SVCLogSource"` (required).

`Directory`

The path to the directory containing the log files (required).

`FileNameFilter`

Optionally limits the set of files in the directory where log data is collected based on a wildcard file-naming pattern. If this key-value pair is not specified, then by default, log data from all files in the directory is collected.

UlsSource Configuration

The `UlsSource` type is used to collect logs from Microsoft SharePoint. The following is an example `UlsSource` declaration:

```
{
    "Id": "UlsSource",
    "SourceType": "UlsSource",
    "Directory": "C:\\temp\\uls",
    "FileNameFilter": "*.log"
}
```

All `UlsSource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string `"UlsSource"` (required).

`Directory`

The path to the directory containing the log files (required).

`FileNameFilter`

Optionally limits the set of files in the directory where log data is collected based on a wildcard file-naming pattern. If this key-value pair is not specified, then by default, log data from all files in the directory is collected.

WindowsEventLogSource Configuration

The `WindowsEventLogSource` type is used to collect events from the Windows Event Log service. The following is an example `WindowsEventLogSource` declaration:

```
{
    "Id": "mySecurityLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Security"
}
```

All `WindowsEventLogSource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string `"WindowsEventLogSource"` (required).

`LogName`

Events are collected from the specified log. Common values include `Application`, `Security`, and `System`, but you can specify any valid Windows event log name. This key-value pair is required.

`Query`

Optionally limits what events are output from the `WindowsEventLogSource`. If this key-value pair is not specified, then by default, all events are output. For information about the syntax of

this value, see Event Queries and Event XML in the Windows documentation. For information about log level definitions, see Event Types in the Windows documentation.

`IncludeEventData`

Optionally enables the collection and streaming of provider-specific event data associated with events from the specified Windows event log when the value of this key-value pair is `"true"`. Only event data that can be successfully serialized is included. This key-value pair is optional, and if it is not specified, the provider-specific event data is not collected.

> **Note**
>
> Including event data could significantly increase the amount of data streamed from this source. The maximum size of an event can be 262,143 bytes with event data included.

The parsed output from the `WindowsEventLogSource` contains the following information:

Attribute	Type	Description
`EventId`	Int	The identifier of the type of event.
`Description`	String	Text that describes the details of the event.
`LevelDisplayName`	String	The category of event (one of Error, Warning, Information, Success Audit, Failure Audit).
`LogName`	String	Where the event was recorded (typical values are `Application` , `Security`, and `System`, but there are many possibilities).
`MachineName`	String	Which computer recorded the event.
`ProviderName`	String	Which application or service recorded the event.

Attribute	Type	Description
`TimeCreated`	String	When the event occurred in ISO 8601 format.
`Index`	Int	Where the entry is located in the log.
`UserName`	String	Who made the entry if known.
`Keywords`	String	The type of event. Standard values include `AuditFailure` (failed security audit events), `AuditSuccess` (successful security audit events), `Classic` (events raised with the `RaiseEvent` function), `Correlation Hint` (transfer events), `SQM` (Service Quality Mechanism events), `WDI Context` (Windows Diagnostic Infrastructure context events), and `WDI Diag` (Windows Diagnostic Infrastructure diagnostics events).
`EventData`	List of objects	Optional provider-specific extra data about the log event. This is only included if the value for the `IncludeEventData` key-value pair is `"true"`.

The following is an example event transformed into JSON:

```
{[
    "EventId": 7036,
    "Description": "The Amazon SSM Agent service entered the stopped state.",
    "LevelDisplayName": "Informational",
    "LogName": "System",
    "MachineName": "mymachine.mycompany.com",
    "ProviderName": "Service Control Manager",
```

```
    "TimeCreated": "2017-10-04T16:42:53.8921205Z",
    "Index": 462335,
    "UserName": null,
    "Keywords": "Classic",
    "EventData": [
    "Amazon SSM Agent",
    "stopped",
    "rPctBAMZFhYubF8zVLcrBd3bTTcNzHvY5Jc2Br0aMrxxx=="
]}
```

WindowsEventLogPollingSource Configuration

`WindowsEventLogPollingSource` uses a polling-based mechanism to gather all new events from the event log that match the configured parameters. The polling interval is updated dynamically between 100 ms and 5000 ms depending on how many events were gathered during the last poll. The following is an example `WindowsEventLogPollingSource` declaration:

```
{
    "Id": "MySecurityLog",
    "SourceType": "WindowsEventLogPollingSource",
    "LogName": "Security",
    "IncludeEventData": "true",
    "Query": "",
    "CustomFilters": "ExcludeOwnSecurityEvents"
}
```

All `WindowsEventLogPollingSource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string "`WindowsEventLogPollingSource`" (required).

`LogName`

Specifies the log. Valid options are `Application`, `Security`, `System`, or other valid logs.

`IncludeEventData`

Optional. When `true`, specifies that extra EventData when streamed as JSON and XML is included. Default is `false`.

`Query`

Optional. Windows event logs support querying events using XPath expressions, which you can specify using `Query`. For more information, see Event Queries and Event XML in Microsoft documentation.

`CustomFilters`

Optional. A list of filters separated by a semi-colon (`;`). The following filters can be specified.

`ExcludeOwnSecurityEvents`

Excludes security events generated by Kinesis Agent for Windows itself.

WindowsETWEventSource Configuration

The `WindowsETWEventSource` type is used to collect application and service event traces using a feature named Event Tracing for Windows (ETW). For more information, see Event Tracing in the Windows documentation.

The following is an example `WindowsETWEventSource` declaration:

```
{
    "Id": "ClrETWEventSource",
    "SourceType": "WindowsETWEventSource",
    "ProviderName": "Microsoft-Windows-DotNETRuntime",
    "TraceLevel": "Verbose",
    "MatchAnyKeyword": 32768
}
```

All `WindowsETWEventSource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string `"WindowsETWEventSource"` (required).

`ProviderName`

Specifies which event provider to use to collect trace events. This must be a valid ETW provider name for an installed provider. To determine which providers are installed, execute the following in a Windows command prompt window:

```
logman query providers
```

`TraceLevel`

Specifies what categories of trace events should be collected. Allowed values include `Critical`, `Error`, `Warning`, `Informational`, and `Verbose`. The exact meaning depends on the ETW provider that is selected.

`MatchAnyKeyword`

This value is a 64-bit number, in which each bit represents an individual keyword. Each keyword describes a category of events to be collected. For the supported keywords and their values and how they related to `TraceLevel`, see the documentation for that provider. For example, for information about the CLR ETW provider, see CLR ETW Keywords and Levels in the Microsoft .NET Framework documentation.

In the previous example, 32768 (0x00008000) represents the `ExceptionKeyword` for the CLR ETW provider that instructs the provider to collect information about exceptions thrown. Although JSON doesn't natively support hex constants, you can specify them for `MatchAnyKeyword` by placing them in a string. You can also specify several constants separated by commas. For example, use the following to specify both the `ExceptionKeyword` and `SecurityKeyword` (0x00000400):

```
{
   "Id": "MyClrETWEventSource",
   "SourceType": "WindowsETWEventSource",
   "ProviderName": "Microsoft-Windows-DotNETRuntime",
   "TraceLevel": "Verbose",
   "MatchAnyKeyword": "0x00008000, 0x00000400"
}
```

To ensure that all specified keywords are enabled for a provider, multiple keyword values are combined using OR and passed to that provider.

The output from the `WindowsETWEventSource` contains the following information for each event:

Attribute	Type	Description
`EventName`	String	What kind of event occurred.

Attribute	Type	Description
ProviderName	String	Which provider detected the event.
FormattedMessage	String	A textual summary of the event.
ProcessID	Int	Which process reported the event.
ExecutingThreadID	Int	Which thread within the process reported the event.
MachineName	String	The name of the desktop or server that is reporting the event.
Payload	Hashtable	A table with a string key and any kind of object as a value. The key is the payload item name, and the value is the payload item's value. The payload is provider dependent.

The following is an example event transformed into JSON:

```
{
     "EventName": "Exception/Start",
     "ProviderName": "Microsoft-Windows-DotNETRuntime",
     "FormattedMessage": "ExceptionType=System.Exception;\r
\nExceptionMessage=Intentionally unhandled exception.;\r\nExceptionEIP=0x2ab0499;\r
\nExceptionHRESULT=-2,146,233,088;\r\nExceptionFlags=CLSCompliant;\r\nClrInstanceID=9
 ",
     "ProcessID": 3328,
     "ExecutingThreadID": 6172,
     "MachineName": "MyHost.MyCompany.com",
     "Payload":
      {
        "ExceptionType": "System.Exception",
        "ExceptionMessage": "Intentionally unhandled exception.",
```

```
        "ExceptionEIP": 44762265,
        "ExceptionHRESULT": -2146233088,
        "ExceptionFlags": 16,
        "ClrInstanceID": 9
      }
}
```

WindowsPerformanceCounterSource Configuration

The `WindowsPerformanceCounterSource` type collects performance counter metrics from Windows. The following is an example `WindowsPerformanceCounterSource` declaration:

```
{
 "Id": "MyPerformanceCounter",
 "SourceType": "WindowsPerformanceCounterSource",
 "Categories": [{
   "Category": "Server",
   "Counters": ["Files Open", "Logon Total", "Logon/sec", "Pool Nonpaged Bytes"]
  },
  {
   "Category": "System",
   "Counters": ["Processes", "Processor Queue Length", "System Up Time"]
  },
  {
   "Category": "LogicalDisk",
   "Instances": "*",
   "Counters": [
    "% Free Space", "Avg. Disk Queue Length",
    {
     "Counter": "Disk Reads/sec",
     "Unit": "Count/Second"
    },
    "Disk Writes/sec"
   ]
  },
  {
   "Category": "Network Adapter",
   "Instances": "^Local Area Connection\* \d$",
   "Counters": ["Bytes Received/sec", "Bytes Sent/sec"]
  }
 ]
}
```

All `WindowsPerformanceCounterSource` declarations can provide the following key-value pairs:

`SourceType`

Must be the literal string `"WindowsPerformanceCounterSource"` (required).

`Categories`

Specifies a set of performance counter metric groups to gather from Windows. Each metric group contains the following key-value pairs:

`Category`

Specifies the counter set of metrics to be collected (required).

`Instances`

Specifies the set of objects of interest when there are a unique set of performance counters per object. For example, when the category is `LogicalDisk`, there are a set of performance counters per disk drive. This key-value pair is optional. You can use the wildcards `*` and `?` to match multiple instances. To aggregate values across all instances, specify `_Total`.

You can also use `InstanceRegex`, which accepts regular expressions that contain the `*` wild card character as part of the instance name.

`Counters`

Specifies which metrics to gather for the specified category. This key-value pair is required. You can use the wildcards `*` and `?` to match multiple counters. You can specify `Counters` using only the name, or by using the name and unit. If counter units are not specified, Kinesis Agent for Windows attempts to infer the units from the name. If those inferences are incorrect, then the unit can be explicitly specified. You can change `Counter` names if you want. The more complex representation of a counter is an object with the following key-value pairs:

`Counter`

The name of the counter. This key-value pair is required.

`Rename`

The name of the counter to present to the sink. This key-value pair is optional.

`Unit`

The meaning of the value that is associated with the counter. For a complete list of valid unit names, see the unit documentation in MetricDatum in the *Amazon CloudWatch API Reference*.

The following is an example of a complex counter specification:

```
{
   "Counter": "Disk Reads/sec,
   "Rename": "Disk Reads per second",
   "Unit": "Count/Second"
}
```

`WindowsPerformanceCounterSource` can only be used with a pipe that specifies an Amazon CloudWatch sink. Use a separate sink if Kinesis Agent for Windows built-in metrics are also streamed to CloudWatch. Examine the Kinesis Agent for Windows log after service startup to determine what units have been inferred for counters when units have not been specified in the `WindowsPerformanceCounterSource` declarations. Use PowerShell to determine the valid names for categories, instances, and counters.

To see information about all categories, including counters associated with counter sets, execute this command in a PowerShell window:

```
Get-Counter -ListSet * | Sort-Object
```

To determine what instances are available for each of the counters in the counter set, execute a command similar to the following example in a PowerShell window:

```
Get-Counter -Counter "\Process(*)\% Processor Time"
```

The value of the `Counter` parameter should be one of the paths from a `PathsWithInstances` member listed by the previous `Get-Counter -ListSet` command invocation.

Kinesis Agent for Windows Built-In Metrics Source

In addition to ordinary metrics sources such as the `WindowsPerformanceCounterSource` type (see WindowsPerformanceCounterSource Configuration), the CloudWatch sink type can receive metrics from a special source that gathers metrics about Kinesis Agent for Windows itself. Kinesis Agent for Windows metrics are also available in the `KinesisTap` category of Windows performance counters.

The `MetricsFilter` key-value pair for the CloudWatch sink declarations specifies which metrics are streamed to CloudWatch from the built-in Kinesis Agent for Windows metrics source. The value is a string that contains one or more filter expressions separated by semicolons; for example:

`"MetricsFilter": "`*FilterExpression1*`;`*FilterExpression2*`"`

A metric that matches one or more filter expressions is streamed to CloudWatch.

Single instance metrics are global in nature and not tied to a particular source or sink. Multiple instance metrics are dimensional based on the source or sink declaration `Id`. Each source or sink type can have a different set of metrics.

For a list of built-in Kinesis Agent for Windows metric names, see List of Kinesis Agent for Windows Metrics.

For single instance metrics, the filter expression is the name of the metric; for example:

```
"MetricsFilter": "SourcesFailedToStart;SinksFailedToStart"
```

For multiple instance metrics, the filter expression is the name of the metric, a period (`.`), and then the `Id` of the source or sink declaration that generated that metric. For example, assuming there is a sink declaration with an `Id` of `MyFirehose`:

```
"MetricsFilter": "KinesisFirehoseRecordsFailedNonrecoverable.MyFirehose"
```

You can use special wildcard patterns that are designed to distinguish between single and multiple instance metrics.

- Asterisk (*) matches zero or more characters except period (`.`).

- Question mark (?) matches one character except period.
- Any other character only matches itself.
- `_Total` is a special token that causes the aggregation of all matching multiple instance values across the dimension.

The following example matches all single instance metrics:

```
"MetricsFilter": "*"
```

Because an asterisk does not match the period character, only single instance metrics are included.

The following example matches all multiple instance metrics:

```
"MetricsFilter": "*.*"
```

The following example matches all metrics (single and multiple):

```
"MetricsFilter": "*;*.*"
```

The following example aggregates all multiple instance metrics across all sources and sinks:

```
"MetricsFilter": "*._Total"
```

The following example aggregates all Firehose metrics for all Firehose sinks:

```
"MetricsFilter": "*Firehose*._Total"
```

The following example matches all single and multiple instance error metrics:

```
"MetricsFilter": "*Failed*;*Error*.*;*Failed*.*"
```

The following example matches all non-recoverable error metrics aggregated across all sources and sinks:

```
"MetricsFilter": "*Nonrecoverable*._Total"
```

For information about how to specify a pipe that uses the Kinesis Agent for Windows built-in metric source, see Configuring Kinesis Agent for Windows Metric Pipes.

List of Kinesis Agent for Windows Metrics

The following is a list of single instance and multiple instance metrics that are available for Kinesis Agent for Windows.

Single Instance Metrics

The following single instance metrics are available:

`KinesisTapBuildNumber`

The version number of Kinesis Agent for Windows.

`PipesConnected`

How many pipes have connected their source to their sink successfully.

`PipesFailedToConnect`

How many pipes have connected their source to their sink unsuccessfully.

`SinkFactoriesFailedToLoad`

How many sink types did not load into Kinesis Agent for Windows successfully.

`SinkFactoriesLoaded`

How many sink types loaded into Kinesis Agent for Windows successfully.

`SinksFailedToStart`

How many sinks did not begin successfully, usually due to incorrect sink declarations.

`SinksStarted`

How many sinks began successfully.

`SourcesFailedToStart`

How many sources did not begin successfully, usually due to incorrect source declarations.

`SourcesStarted`

How many sources began successfully.

`SourceFactoriesFailedToLoad`

How many source types did not load into Kinesis Agent for Windows successfully.

`SourceFactoriesLoaded`

How many source types loaded successfully into Kinesis Agent for Windows.

Multiple Instance Metrics

The following multiple instance metrics are available:

DirectorySource Metrics

`DirectorySourceBytesRead`

How many bytes were read during the interval for this `DirectorySource`.

`DirectorySourceBytesToRead`

How many known numbers of bytes are available to read that have not been read yet by Kinesis Agent for Windows.

`DirectorySourceFilesToProcess`

How many known files to examine that have not yet been examined yet by Kinesis Agent for Windows.

`DirectorySourceRecordsRead`

How many records have been read during the interval for this `DirectorySource`.

WindowsEventLogSource Metrics

`EventLogSourceEventsError`

How many Windows event log events were not read successfully.

`EventLogSourceEventsRead`

How many Windows event log events were read successfully.

KinesisFirehose Sink Metrics

`KinesisFirehoseBytesAccepted`

How many bytes were accepted during the interval.

`KinesisFirehoseClientLatency`

How much time passed between record generation and record streaming to the Firehose service.

`KinesisFirehoseLatency`

How much time passed between the start and end of record streaming for the Firehose service.

`KinesisFirehoseNonrecoverableServiceErrors`

How many times records could not be sent without error to the Firehose service despite retries.

`KinesisFirehoseRecordsAttempted`

How many records tried to be streamed to the Firehose service.

`KinesisFirehoseRecordsFailedNonrecoverable`

How many records were not successfully streamed to the Firehose service despite retries.

`KinesisFirehoseRecordsFailedRecoverable`

How many records were successfully streamed to the Firehose service, but only with retries.

`KinesisFirehoseRecordsSuccess`

How many records were successfully streamed to the Firehose service without retries.

`KinesisFirehoseRecoverableServiceErrors`

How many times records could successfully be sent to the Firehose service, but only with retries.

KinesisStream Metrics

`KinesisStreamBytesAccepted`

How many bytes were accepted during the interval.

`KinesisStreamClientLatency`

How much time passed between record generation and record streaming to the Kinesis Data Streams service.

`KinesisStreamLatency`

How much time passed between the start and end of record streaming for the Kinesis Data Streams service.

`KinesisStreamNonrecoverableServiceErrors`

How many times records could not be sent without error to the Kinesis Data Streams service despite retries.

`KinesisStreamRecordsAttempted`

How many records tried to be streamed to the Kinesis Data Streams service.

`KinesisStreamRecordsFailedNonrecoverable`

How many records were not successfully streamed to the Kinesis Data Streams service despite retries.

`KinesisStreamRecordsFailedRecoverable`

How many records were successfully streamed to the Kinesis Data Streams service, but only with retries.

`KinesisStreamRecordsSuccess`

How many records were successfully streamed to the Kinesis Data Streams service without retries.

`KinesisStreamRecoverableServiceErrors`

How many times records could successfully be sent to the Kinesis Data Streams service, but only with retries.

CloudWatchLog Metrics

`CloudWatchLogBytesAccepted`

How many bytes were accepted during the interval.

`CloudWatchLogClientLatency`

How much time passed between record generation and record streaming to the CloudWatch Logs service.

`CloudWatchLogLatency`

How much time passed between the start and end of record streaming for the CloudWatch Logs service.

`CloudWatchLogNonrecoverableServiceErrors`

How many times records could not be sent without error to the CloudWatch Logs service despite retries.

`CloudWatchLogRecordsAttempted`

How many records tried to be streamed to the CloudWatch Logs service.

`CloudWatchLogRecordsFailedNonrecoverable`

How many records were not successfully streamed to the CloudWatch Logs service despite retries.

`CloudWatchLogRecordsFailedRecoverable`

How many records were successfully streamed to the CloudWatch Logs service, but only with retries.

`CloudWatchLogRecordsSuccess`

How many records were successfully streamed to the CloudWatch Logs service without retries.

`CloudWatchLogRecoverableServiceErrors`

How many times records could successfully be sent to the CloudWatch Logs service, but only with retries.

CloudWatch Metrics

`CloudWatchLatency`

How much time on average passed between the start and end of metric streaming for the CloudWatch service.

`CloudWatchNonrecoverableServiceErrors`

How many times metrics could not be sent without error to the CloudWatch service despite retries.

`CloudWatchRecoverableServiceErrors`

How many times metrics were sent without error to the CloudWatch service but only with retries.

`CloudWatchServiceSuccess`

How many times metrics were sent without error to the CloudWatch service with no retries needed.

Bookmark Configuration

By default, Kinesis Agent for Windows sends log records to sinks that are created after the agent starts. Sometimes it is useful to send earlier log records, for example, log records that are created during the time period when Kinesis Agent for Windows stops during an automatic update. The bookmark feature tracks what records have been sent to sinks. When Kinesis Agent for Windows is in bookmark mode and starts up, it sends all log records that were created after Kinesis Agent for Windows stopped, along with any subsequently created log records. To control this behavior, file-based source declarations can optionally include the following key-value pairs:

`InitialPosition`

Specifies the initial situation for the bookmark. Possible values are as follows:

`EOS`

Specifies end of stream (EOS). Only log records created while the agent is running are sent to sinks.

`0`

All available log records and events are initially sent. Then a bookmark is created to ensure that every new log record and event created after the bookmark was created are eventually sent, whether or not Kinesis Agent for Windows is running.

`Bookmark`

The bookmark is initialized to just after the latest log record or event. Then a bookmark is created to ensure that every new log record and event created after the bookmark was created are eventually sent, whether or not Kinesis Agent for Windows is running.

Bookmarks are enabled by default. Files are stored in the `%ProgramData%\Amazon\KinesisTap` directory.

`Timestamp`

Log records and events that are created after the `InitialPositionTimestamp` value (definition follows) are sent. Then a bookmark is created to ensure that every new log record

and event created after the bookmark was created are eventually sent whether or not Kinesis Agent for Windows is running.

`InitialPositionTimestamp`

Specifies the earliest log record or event timestamp that you want. Specify this key-value pair only when `InitialPosition` has a value of `Timestamp`.

`BookmarkOnBufferFlush`

This setting can be added to any bookmarkable source. When set to `true`, ensures that bookmark updates occur only when a sink successfully ships an event to AWS. You can only subscribe a single sink to a source. If you are shipping logs to multiple destinations, duplicate your sources to avoid potential issues with data loss.

When Kinesis Agent for Windows has been stopped for a long time, it might be necessary to delete those bookmarks because log records and events that are bookmarked might no longer exist. Bookmark files for a given *source id* are located in `%PROGRAMDATA%\Amazon\AWSKinesisTap\`*`source id`*`.bm`.

Bookmarks do not work on files that are renamed or truncated. Because of the nature of ETW events and performance counters, they cannot be bookmarked.

Sink Declarations

Sink declarations specify where and in what form logs, events, and metrics should be sent to various AWS services. The following sections describe configurations for the built-in sink types that are available in Amazon Kinesis Agent for Microsoft Windows. Because Kinesis Agent for Windows is extensible, you can add custom sink types. Each sink type typically requires unique key-value pairs in the configuration declarations that are relevant for that sink type.

All sink declarations can contain the following key-value pairs:

`Id`

A unique string that identifies a particular sink within the configuration file (required).

`SinkType`

The name of the sink type for this sink (required). The sink type specifies the destination of the log, event, or metric data that is being streamed by this sink.

`AccessKey`

Specifies the AWS access key to use when authorizing access to the AWS service that is associated with the sink type. This key-value pair is optional. For more information, see Sink Security Configuration.

`SecretKey`

Specifies the AWS secret key to use when authorizing access to the AWS service that is associated with the sink type. This key-value pair is optional. For more information, see Sink Security Configuration.

`Region`

Specifies which AWS Region contains the destination resources for streaming. This key-value pair is optional.

`ProfileName`

Specifies which AWS profile to use for authentication. This key-value pair is optional, but if specified, it overrides any specified access key and secret key. For more information, see Sink Security Configuration.

`RoleARN`

Specifies the IAM role to use when accessing the AWS service that is associated with the sink type. This option is useful when Kinesis Agent for Windows is running on an EC2 instance but a different role would be more appropriate than the role referenced by the instance profile. For example, a cross-account role can be used to target resources that are not in the same AWS account as the EC2 instance. This key-value pair is optional.

`Format`

Specifies the kind of serialization that is applied to logs and event data before streaming. Valid values are `json` and `xml`. This option is helpful when downstream analytics in the data pipeline require or prefer data in a particular form. This key-value pair is optional, and if not specified, ordinary text from the source is streamed from the sink to the AWS service that is associated with the sink type.

`TextDecoration`

When no `Format` is specified, `TextDecoration` specifies what additional text should be included when streaming log or event records. For more information, see Configuring Sink Decorations. This key-value pair is optional.

`ObjectDecoration`

When `Format` is specified, `ObjectDecoration` specifies what additional data is included in the log or event record before serialization and streaming. For more information, see Configuring Sink Decorations. This key-value pair is optional.

`BufferInterval`

To minimize API calls to the AWS service that is associated with the sink type, Kinesis Agent for Windows buffers up multiple log, event, or metric records before streaming. This can save money for services that charge per API call. `BufferInterval` specifies the maximum length of time (in seconds) that records should be buffered before streaming to the AWS service. This key-value pair is optional, and if specified, use a string to represent the value.

`BufferSize`

To minimize API calls to the AWS service that is associated with the sink type, Kinesis Agent for Windows buffers up multiple log, event, or metric records before streaming. This can save money for services that charge per API call. `BufferSize` specifies the maximum number of records to buffer before streaming to the AWS service. This key-value pair is optional, and if it is specified, use a string to represent the value.

`MaxAttempts`

Specifies the maximum number of times Kinesis Agent for Windows tries to stream a set of log, event, and metric records to an AWS service if the streaming consistently fails. This key-value pair is optional. If it is specified, use a string to represent the value. The default value is "3".

For examples of complete configuration files that use various kinds of sinks, see Streaming from the Windows Application Event Log to Sinks.

Topics

- Configuring Sink Decorations
- Configuring Sink Variable Substitutions
- Configuring Sink Queuing
- Configuring a Proxy for Sinks
- Configuring resolving variables in more sink attributes
- Configuring AWS STS Regional Endpoints When Using RoleARN Property in AWS Sinks
- Configuring VPC Endpoint for AWS Sinks
- Configuring An Alternate Means of Proxy

`KinesisStream` Sink Configuration

The `KinesisStream` sink type streams log records and events to the Kinesis Data Streams service. Typically, data that is streamed to Kinesis Data Streams is processed by one or more custom applications that execute using various AWS services. Data is streamed to a named stream that is configured using Kinesis Data Streams. For more information, see the *Amazon Kinesis Data Streams Developer Guide*.

The following is an example Kinesis Data Streams sink declaration:

```
{
    "Id": "TestKinesisStreamSink",
    "SinkType": "KinesisStream",
    "StreamName": "MyTestStream",
    "Region": "us-west-2"
}
```

All `KinesisStream` sink declarations can provide the following additional key-value pairs:

`SinkType`

Must be specified, and the value must be the literal string `KinesisStream`.

`StreamName`

Specifies the name of the Kinesis data stream that receives the data streamed from the `KinesisStream` sink type (required). Before streaming the data, configure the stream in the

AWS Management Console, the AWS CLI, or through an application using the Kinesis Data Streams API.

`RecordsPerSecond`

Specifies the maximum number of records streamed to Kinesis Data Streams per second. This key-value pair is optional. If it is specified, use an integer to represent the value. The default value is 1000 records.

`BytesPerSecond`

Specifies the maximum number of bytes streamed to Kinesis Data Streams per second. This key-value pair is optional. If it is specified, use an integer to represent the value. The default value is 1 MB.

The default `BufferInterval` for this sink type is 1 second, and the default `BufferSize` is 500 records.

`KinesisFirehose` Sink Configuration

The `KinesisFirehose` sink type streams log records and events to the Firehose service. Firehose delivers the streamed data to other services for storage. Typically the stored data is then analyzed in subsequent stages of the data pipeline. Data is streamed to a named delivery stream that is configured using Firehose. For more information, see the *Amazon Data Firehose Developer Guide*.

The following is an example Firehose sink declaration:

```
{
   "Id": "TestKinesisFirehoseSink",
   "SinkType": "KinesisFirehose",
   "StreamName": "MyTestFirehoseDeliveryStream",
   "Region": "us-east-1",
   "CombineRecords": "true"
}
```

All `KinesisFirehose` sink declarations can provide the following additional key-value pairs:

`SinkType`

Must be specified, and the value must be the literal string `KinesisFirehose`.

`StreamName`

Specifies the name of the Firehose delivery stream that receives the data streamed from the `KinesisStream` sink type (required). Before streaming the data, configure the delivery stream using the AWS Management Console, the AWS CLI, or through an application using the Firehose API.

`CombineRecords`

When set to `true`, specifies to combine multiple small records into a large record with a 5 KB maximum size. This key-value pair is optional. Records combined using this function are separated by \n. If you use AWS Lambda to transform a Firehose record, your Lambda function needs to account for the separator character.

`RecordsPerSecond`

Specifies the maximum number of records that are streamed to Kinesis Data Streams per second. This key-value pair is optional. If it is specified, use an integer to represent the value. The default value is 5000 records.

`BytesPerSecond`

Specifies the maximum number of bytes that are streamed to Kinesis Data Streams per second. This key-value pair is optional. If it is specified, use an integer to represent the value. The default value is 5 MB.

The default `BufferInterval` for this sink type is 1 second, and the default `BufferSize` is 500 records.

CloudWatch Sink Configuration

The `CloudWatch` sink type streams metrics to the CloudWatch service. You can view the metrics in the AWS Management Console. For more information, see the *Amazon CloudWatch User Guide*.

The following is an example `CloudWatch` sink declaration:

```
{
   "Id": "CloudWatchSink",
   "SinkType": "CloudWatch"
}
```

All `CloudWatch` sink declarations can provide the following additional key-value pairs:

`SinkType`

Must be specified, and the value must be the literal string `CloudWatch`.

`Interval`

Specifies how frequently (in seconds) Kinesis Agent for Windows reports metrics to the CloudWatch service. This key-value pair is optional. If it is specified, use an integer to represent the value. The default value is 60 seconds. Specify 1 second if you want high-resolution CloudWatch metrics.

`Namespace`

Specifies the CloudWatch namespace where the metric data is reported. CloudWatch namespaces group a set of metrics together. This key-value pair is optional. The default value is `KinesisTap`.

`Dimensions`

Specifies the CloudWatch dimensions that are used to isolate metric sets within a namespace. This can be useful to provide separate sets of metric data for each desktop or server, for example. This key-value pair is optional, and if specified, the value must comply with the following format: "*key1=value1*;*key2=value2...*". The default value is `"ComputerName={computername};InstanceId={instance_id}"`. This value supports sink variable substitution. For more information, see Configuring Sink Variable Substitutions.

`MetricsFilter`

Specifies which metrics are streamed to CloudWatch from the built-in Kinesis Agent for Windows metrics source. For more information about the built-in Kinesis Agent for Windows metrics source, including the details of the syntax of the value of this key-value pair, see Kinesis Agent for Windows Built-In Metrics Source.

`CloudWatchLogs` Sink Configuration

The `CloudWatchLogs` sink type streams log records and events to Amazon CloudWatch Logs. You can view logs in the AWS Management Console, or process them via additional stages of a data pipeline. Data is streamed to a named log stream that is configured in CloudWatch Logs. Log streams are organized into named log groups. For more information, see the *Amazon CloudWatch Logs User Guide*.

The following is an example CloudWatch Logs sink declaration:

```
{
   "Id": "MyCloudWatchLogsSink",
   "SinkType": "CloudWatchLogs",
   "BufferInterval": "60",
   "BufferSize": "100",
   "Region": "us-west-2",
   "LogGroup": "MyTestLogGroup",
   "LogStream": "MyTestStream"
}
```

All `CloudWatchLogs` sink declarations must provide the following additional key-value pairs:

`SinkType`

Must be the literal string `CloudWatchLogs`.

`LogGroup`

Specifies the name of the CloudWatch Logs log group that contains the log stream that receives the log and event records streamed by the `CloudWatchLogs` sink type. If the specified log group does not exist, Kinesis Agent for Windows attempts to create it.

`LogStream`

Specifies the name of the CloudWatch Logs log stream that receives the log and event records stream by the `CloudWatchLogs` sink type. This value supports sink variable substitution. For more information, see Configuring Sink Variable Substitutions. If the specified log stream does not exist, Kinesis Agent for Windows attempts to create it.

The default `BufferInterval` for this sink type is 1 second, and the default `BufferSize` is 500 records. The maximum buffer size is 10,000 records.

Local `FileSystem` Sink Configuration

The sink type `FileSystem` saves log and event records to a file on the local file system instead of streaming them to AWS services. `FileSystem` sinks are useful for testing and diagnostics. For example, you can use this sink type to examine records before sending them to AWS.

With `FileSystem` sinks, you can also use configuration parameters to simulate batching, throttling, and retry-on-error to mimic the behavior of actual AWS sinks.

All records from all sources connected to a `FileSystem` sink are saved to the single file specified as `FilePath`. If `FilePath` is not specified, records are saved to a file named *`SinkId`*`.txt` in the `%TEMP%` directory, which is usually `C:\Users\`*`UserName`*`\AppData\Local\Temp`, where *`SinkId`* is the unique identifier of the sink and *`UserName`* is the Windows user name of the active user.

This sink type supports text decoration attributes. For more information, see Configuring Sink Decorations.

An example `FileSystem` sink type configuration is shown in the following example.

```
{
    "Id": "LocalFileSink",
    "SinkType": "FileSystem",
    "FilePath": "C:\\ProgramData\\Amazon\\local_sink.txt",
    "Format": "json",
    "TextDecoration": "",
    "ObjectDecoration": ""
}
```

The `FileSystem` configuration consists of the following key-value pairs.

`SinkType`

Must be the literal string `FileSystem`.

`FilePath`

Specifies the path and file where records are saved. This key-value pair is optional. If not specified, the default is *`TempPath`*`\\`*`SinkId`*`.txt`, where *`TempPath`* is the folder stored in the `%TEMP%` variable and *`SinkId`* is the unique identifier of the sink.

`Format`

Specifies the format of the event to be `json` or `xml`. This key value pair is optional and case-insensitive. If omitted, events are written to the file in plain text.

`TextDecoration`

Applies only to events written in plain text. This key-value pair is optional.

`ObjectDecoration`

Applies only to events where `Format` is set to `json`. This key-value pair is optional.

Advanced Usage – Record Throttling and Failure Simulation

`FileSystem` can mimic the behavior of AWS sinks by simulating record throttling. You can use the following key-value pairs to specify record throttling and failure simulation attributes.

By acquiring a lock on the destination file and preventing writes to it, you can use `FileSystem` sinks to simulate and examine the behavior of AWS sinks when the network fails.

The following example shows a `FileSystem` configuration with simulation attributes.

```
{
    "Id": "LocalFileSink",
    "SinkType": "FileSystem",
    "FilePath": "C:\\ProgramData\\Amazon\\local_sink.txt",
    "TextDecoration": "",
    "RequestsPerSecond": "100",
    "BufferSize": "10",
    "MaxBatchSize": "1024"
}
```

`RequestsPerSecond`

Optional and specified as a string type. If omitted, the default is `"5"`. Controls the rate of requests that the sink processes—that is, writes to file—not the number of records. Kinesis Agent for Windows makes batch requests to AWS endpoints, so a request may contain multiple records.

`BufferSize`

Optional and specified as string type. Specifies the maximum number of event records that the sink batches before saving to file.

`MaxBatchSize`

Optional and specified as a string type. Specifies the maximum amount of event record data in bytes that the sink batches before saving to file.

The maximum record rate limit is a function of `BufferSize`, which determines the maximum number of records per request, and `RequestsPerSecond`. You can calculate the record rate limit per second using the following formula.

RecordRate = `BufferSize` * `RequestsPerSecond`

Given configuration values in the example above, there is a maximum record rate of 1000 records per second.

Sink Security Configuration

Configuring Authentication

For Kinesis Agent for Windows to stream logs, events, and metrics to AWS services, access must be authenticated. There are several ways to provide authentication for Kinesis Agent for Windows. How you do it depends on the situation where Kinesis Agent for Windows is executing and the specific security requirements for a particular organization.

- If Kinesis Agent for Windows is executing on an Amazon EC2 host, the most secure and simplest way to provide authentication is to create an IAM role with sufficient access to the required operations for the required AWS services, and an EC2 instance profile that references that role. For information about creating instance profiles, see Using Instance Profiles. For information about what policies to attach to the IAM role, see Configuring Authorization.

  After creating the instance profile, you can associate it with any EC2 instances that use Kinesis Agent for Windows. If instances already have an associated instance profile, you can attach the appropriate policies to the role that is associated with that instance profile.
- If Kinesis Agent for Windows executes on an EC2 host in one account, but the resources that are the target of the sink reside in a different account, you can create an IAM role for cross-account access. For more information, see Tutorial: Delegate Access Across AWS accounts Using IAM Roles. After creating the cross-account role, specify the Amazon Resource Name (ARN) for the cross-account role as the value of the `RoleARN` key-value pair in the sink declaration. Kinesis Agent for Windows then attempts to assume the specified cross-account role when accessing AWS resources that are associated with the sink type for that sink.
- If Kinesis Agent for Windows is executing outside of Amazon EC2 (for example, on-premises), several options exist:
  - If it is acceptable to register the on-premises server or desktop machine as an Amazon EC2 Systems Manager managed-instance, use the following process to configure authentication:
    1. Use the process described in Setting Up AWS Systems Manager in Hybrid Environments to create a service role, create an activation for a managed instance, and install the SSM agent.
    2. Attach the appropriate policies to the service role to enable Kinesis Agent for Windows to access the resources necessary for streaming data from the configured sinks. For information about what policies to attach to the IAM role, see Configuring Authorization.

3. Use the process described in Configuring `ProfileRefreshingAWSCredentialProvider` to Refresh AWS Credentials to refresh AWS credentials.

   This is the recommended approach for non-EC2 instances because credentials are securely managed by SSM and AWS.

- If it's acceptable to run the `AWSKinesisTap` service for Kinesis Agent for Windows under a specific user instead of the default system account, use the following process:

  1. Create an IAM user in the AWS account where the AWS services will be used. Capture the access key and secret key of this user during the creation process. You need this information for later steps in this process.
  2. Attach policies to the IAM user that authorize access to the required operations for the required services. For information about what policies to attach to the IAM user, see Configuring Authorization.
  3. Change the `AWSKinesisTap` service on each desktop or server so that it runs under a specific user rather than the default system account.
  4. Create a profile in the SDK store using the access key and secret key recorded earlier. For more information, see Configuring AWS Credentials.
  5. Update the `AWSKinesisTap.exe.config` file in the `%PROGRAMFILES%\Amazon\AWSKinesisTap` directory to specify the name of the profile created in the previous step. For more information, see Configuring AWS Credentials.

  This is the recommended approach for non-EC2 hosts that cannot be managed instances because the credentials are encrypted for the specific host and the specific user.

- If it is required to run the `AWSKinesisTap` service for Kinesis Agent for Windows under the default system account, you must use a shared credential file. This is because the system account has no Windows user profile for enabling the SDK store. Shared credential files are not encrypted, so we do not recommend this approach. For information about how to use shared configuration files, see Configuring AWS Credentials in the *AWS SDK for .NET*. If you use this approach, we recommend that you use NTFS encryption and restricted file access to the shared configuration file. Keys should be rotated by a management platform, and the shared configuration file must be updated when the key rotation occurs.

Although it is possible to directly provide access keys and secret keys in the sink declarations, this approach is discouraged because the declarations are not encrypted.

Configuring Authorization

Attach the appropriate policies that follow to the IAM user or role that Kinesis Agent for Windows will use to stream data to AWS services:

Kinesis Data Streams

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "arn:aws:kinesis:*:*:stream/*"
        }
    ]
}
```

To limit authorization to a specific Region, account, or stream name, replace the appropriate asterisks in the ARN with specific values. For more information, see "Amazon Resource Names (ARNs) for Kinesis Data Streams" in Controlling Access to Amazon Kinesis Data Streams Resources Using IAM.

Firehose

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
```

```
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/*"
        }
    ]
}
```

To limit authorization to a specific Region, account, or delivery stream name, replace the appropriate asterisks in the ARN with specific values. For more information, see Controlling Access with Amazon Kinesis Data Firehose in the *Amazon Data Firehose Developer Guide*.

CloudWatch

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        }
    ]
}
```

For more information, see Overview of Managing Access Permissions to Your CloudWatch Resources in the *Amazon CloudWatch Logs User Guide*.

CloudWatch Logs with an Existing Log Group and Log Stream

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*"
        },
        {
            "Sid": "VisualEditor4",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:*:*:*"
        }
    ]
}
```

To restrict access to a specific Region, account, log group, or log stream, replace the appropriate asterisks in the ARNs with appropriate values. For more information, see Overview of Managing Access Permissions to Your CloudWatch Logs Resources in the *Amazon CloudWatch Logs User Guide*.

CloudWatch Logs with Extra Permissions for Kinesis Agent for Windows to Create Log Groups and Log Streams

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor5",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
```

```
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*"
        },
        {
            "Sid": "VisualEditor6",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:*:*:*"
        },
        {
            "Sid": "VisualEditor7",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        }
    ]
}
```

To restrict access to a specific Region, account, log group, or log stream, replace the appropriate asterisks in the ARNs with appropriate values. For more information, see Overview of Managing Access Permissions to Your CloudWatch Logs Resources in the *Amazon CloudWatch Logs User Guide*.

Permissions Required for EC2 Tag Variable Expansion

Using variable expansion with the `ec2tag` variable prefix requires the `ec2:Describe*` permission.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [{
      "Sid": "VisualEditor8",
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    }
   ]
}
```

> **Note**
>
> You can combine multiple statements into a single policy as long as the `Sid` for each statement is unique within that policy. For information about creating policies, see Creating IAM Policies in the *IAM User Guide*.

Configuring `ProfileRefreshingAWSCredentialProvider` to Refresh AWS Credentials

If you use AWS Systems Manager for hybrid environments to manage AWS credentials, Systems Manager rotates session credentials in `c:\Windows\System32\config\systemprofile\.aws\credentials`. For more information about Systems Manager for hybrid environments, see Setting up AWS Systems Manager for hybrid environments in the *AWS Systems Manager User Guide*.

Because the AWS .net SDK does not pick up new credentials automatically, we provide the `ProfileRefreshingAWSCredentialProvider` plug-in to refresh credentials.

You can use the `CredentialRef` attribute of any AWS sync configuration to reference a `Credentials` definition where the `CredentialType` attribute is set to `ProfileRefreshingAWSCredentialProvider` as shown in the following example.

```
{
    "Sinks": [{
        "Id": "myCloudWatchLogsSink",
        "SinkType": "CloudWatchLogs",
        "CredentialRef": "ssmcred",
        "Region": "us-west-2",
        "LogGroup": "myLogGroup",
        "LogStream": "myLogStream"
    }],
    "Credentials": [{
        "Id": "ssmcred",
        "CredentialType": "ProfileRefreshingAWSCredentialProvider",
        "Profile": "default",
        "FilePath": "%USERPROFILE%//.aws//credentials",
        "RefreshingInterval": 300
    }]
}
```

A credential definition consists of the following attributes as key-value pairs.

`Id`

Defines the string that sink definitions can specify using `CredentialRef` to reference this credential configuration.

`CredentialType`

Set to the literal string `ProfileRefreshingAWSCredentialProvider`.

`Profile`

Optional. The default is `default`.

`FilePath`

Optional. Specifies the path to the AWS credentials file. If omitted, `%USERPROFILE%/.aws/credentials` is the default.

`RefreshingInterval`

Optional. The frequency at which credentials are refreshed, in seconds. If omitted, `300` is the default.

Configuring Sink Decorations

Sink declarations can optionally include key-value pairs that specify additional data to stream to various AWS services to enhance the records gathered from the source.

`TextDecoration`

Use this key-value pair when no `Format` is specified in the sink declaration. The value is a special format string where variable substitution occurs. For example, suppose that a `TextDecoration` of `"{ComputerName}:::{timestamp:yyyy-MM-dd HH:mm:ss}:::{_record}"` is provided for a sink. When a source emits a log record that contains the text `The system has resumed from sleep.`, and that source is connected to the sink via a pipe, then the text `MyComputer1:::2017-10-26 06:14:22:::The system has resumed from sleep.` is streamed to the AWS service associated with the sink type. The `{_record}` variable references the original text record delivered by the source.

`ObjectDecoration`

Use this key-value pair when `Format` is specified in the sink declaration to add additional data before record serialization. For example, suppose that an `ObjectDecoration` of `"ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd HH:mm:ss}"` is provided for a sink that specifies JSON `Format`. The resulting JSON streamed to the AWS service associated with the sink type includes the following key-value pairs in addition to the original data from the source:

```
{
    ComputerName: "MyComputer2",
    DT: "2017-10-17 21:09:04"
}
```

For an example of using `ObjectDecoration`, see Tutorial: Stream JSON Log Files to Amazon S3 Using Kinesis Agent for Windows.

`ObjectDecorationEx`

Specifies an expression, which allows for more flexible data extraction and formatting as compared to `ObjectDecoration`. This field can be used when the format of the sink is `json`. The expression syntax is shown in the following.

```
"ObjectDecorationEx":
 "attribute1={expression1};attribute2={expression2};attribute3={expression3}(;...)"
```

For example, the following `ObjectDecorationEx` attribute:

```
"ObjectDecorationEx":
 "host={env:ComputerName};message={upper(_record)};time={format(_timestamp,
 'yyyyMMdd')}"
```

Transforms the literal record:

`System log message`

Into a JSON object as follows, with the values returned by the expressions:

```
{
    "host": "EC2AMAZ-1234",
    "message": "SYSTEM LOG MESSAGE",
```

```
    "time": "20210201"
}
```

For more information about formulating expressions, see Tips for Writing Expressions. Most of the `ObjectDecoration` declaration should work using the new syntax with the exception of timestamp variables. A `{timestamp:yyyyMMdd}` field in `ObjectDecoration` is expressed as `{format(_timestamp,'yyyyMMdd')}` in `ObjectDecorationEx`.

`TextDecorationEx`

Specifies an expression, which allows for more flexible data extraction and formatting as compared to `TextDecoration`, as shown in the following example.

```
"TextDecorationEx": "Message '{lower(_record)}' at {format(_timestamp, 'yyyy-MM-
dd')}"
```

You can use `TextDecorationEx` to compose JSON objects. Use '@{' to escape open curly brace, as shown in the following example.

```
"TextDecorationEx": "@{ \"var\": \"{upper($myvar1)}\" }"
```

If the source type of the source connected to the sink is `DirectorySource`, then the sink can use three additional variables:

`_FilePath`

The full path to the log file.

`_FileName`

The file name and file name extension of the file.

`_Position`

An integer that represents where the record is located in the log file.

These variables are useful when you use a source that gathers log records from multiple files connected to a sink that streams all the records to a single stream. Injecting the values of these variables into the streaming records enables downstream analytics in the data pipeline to order the records by file and by location within each file.

Tips for Writing Expressions

An expression can be any of the following:

- A variable expression.
- A constant expression, for example, `'hello'`, `1`, `1.21`, `null`, `true`, `false`.
- An invocation expression that calls a function, as shown in the following example.

```
regexp_extract('Info: MID 118667291 ICID 197973259 RID 0 To: <jd@acme.com>', 'To: (\\
\\S+)', 1)
```

Special Characters

Two backslashes are required to escape special characters.

Nesting

Function invocations can be nested, as shown in the following example.

```
format(date(2018, 11, 28), 'MMddyyyy')
```

Variables

There are three types of variables: local, meta, and global.

- **Local variables** start with a $ such as `$message`. They are used to resolve the property of the event object, an entry if the event is a dictionary, or an attribute if the event is a JSON object. If the local variable contains space or special characters, use a quoted local variable such as `$'date created'`.
- **Meta variables** start with an underscore (_) and are used to resolve to the metadata of the event. All event types support the following meta variables.

  `_timestamp`

  The time stamp of the event.

  `_record`

  The raw text representation of the event.

  Log events support the following additional meta variables.

`_filepath`

`_filename`

`_position`

`_linenumber`

- **Global variables** resolve to environment variables, EC2 instance metadata, or EC2tag. For better performance, we recommend that you use the prefix to limit search scope, such as `{env:ComputerName}`, `{ec2:InstanceId}`, and `{ec2tag:Name}`.

Built-in Functions

Kinesis Agent for Windows supports the following built-in functions. If any of the arguments are `NULL` and the function is not designed to handle `NULL`, a `NULL` object is returned.

```
//string functions
int length(string input)
string lower(string input)
string lpad(string input, int size, string padstring)
string ltrim(string input)
string rpad(string input, int size, string padstring)
string rtrim(string input)
string substr(string input, int start)
string substr(string input, int start, int length)
string trim(string input)
string upper(string str)

//regular expression functions
string regexp_extract(string input, string pattern)
string regexp_extract(string input, string pattern, int group)

//date functions
DateTime date(int year, int month, int day)
DateTime date(int year, int month, int day, int hour, int minute, int second)
DateTime date(int year, int month, int day, int hour, int minute, int second, int
 millisecond)

//conversion functions
```

```
int? parse_int(string input)
decimal? parse_decimal(string input)
DateTime? parse_date(string input, string format)
string format(object o, string format)

//coalesce functions
object coalesce(object obj1, object obj2)
object coalesce(object obj1, object obj2, object obj3)
object coalesce(object obj1, object obj2, object obj3, object obj4)
object coalesce(object obj1, object obj2, object obj3, object obj4, object obj5)
object coalesce(object obj1, object obj2, object obj3, object obj4, object obj5, object
 obj6)
```

Configuring Sink Variable Substitutions

The `KinesisStream`, `KinesisFirehose`, and `CloudWatchLogs` sink declarations require either a `LogStream` or `StreamName` key-value pair. The value of these key-values can contain variable references that are automatically resolved by Kinesis Agent for Windows. For `CloudWatchLogs`, the `LogGroup` key-value pair is also required and can contain variables references that are automatically resolved by Kinesis Agent for Windows. The variables are specified using the template {*`prefix:variablename`*} where *`prefix:`* is optional. The supported prefixes are as follows:

- `env` — The variable reference is resolved to the value of the environment variable of the same name.
- `ec2` — The variable reference is resolved to the EC2 instance metadata of the same name.
- `ec2tag` — The variable reference is resolved to the value of the EC2 instance tag of the same name. The `ec2:Describe*` permission is required to access instance tags. For more information, see Permissions Required for EC2 Tag Variable Expansion.

If the prefix isn't specified, if there is an environment variable with the same name as `variablename`, the variable reference is resolved to the value of the environment variable. Otherwise, if `variablename` is `instance_id` or `hostname`, the variable reference is resolved to the value of the EC2 metadata of the same name. Otherwise, the variable reference is not resolved.

The following are examples of valid key-value pairs using variable references:

```
"LogStream": "LogStream_{instance_id}"
```

```
"LogStream": "LogStream_{hostname}"
"LogStream": "LogStream_{ec2:local-hostname}"
"LogStream": "LogStream_{computername}"
"LogStream": "LogStream_{env:computername}"
```

The `CloudWatchLogs` sink declarations support a special format timestamp variable that allows the timestamp of the original log or event record from the source to alter the name of the log stream. The format is `{timestamp:timeformat}`. See the following example:

```
"LogStream": "LogStream_{timestamp:yyyyMMdd}"
```

If the log or event record was generated on June 5, 2017, the value of the `LogStream` key-value pair in the previous example would resolve to `"LogStream_20170605"`.

If authorized, the `CloudWatchLogs` sink type can automatically create new log streams when required based on the generated names. You cannot do this for other sink types because they require additional configuration beyond the name of the stream.

There are special variable substitutions that occur in text and object decoration. For more information, see Configuring Sink Decorations.

Configuring Sink Queuing

The `KinesisStream`, `KinesisFirehose`, and `CloudWatchLogs` sink declarations can optionally enable queuing of records that have failed to stream to the AWS service associated with those sink types due to transient connectivity issues. To enable queuing and automatic streaming retries when connectivity is restored, use the following key-value pairs in the sink declarations:

`QueueType`

Specifies the kind of queuing mechanism to use. The only supported value is `file`, which indicates that records should be queued up in a file. This key-value pair is required in order to enable the queuing feature of Kinesis Agent for Windows. If it is not specified, the default behavior is to queue in memory only, and fail to stream when in memory queueing limits are reached.

`QueuePath`

Specifies the path to the folder that contains the files of queued records. This key-value pair is optional. The default value is `%PROGRAMDATA%\KinesisTap\Queue\`*SinkId* where *SinkId* is the identifier you assigned as the value of the `Id` for the sink declaration.

`QueueMaxBatches`

Limits the total amount of space that Kinesis Agent for Windows can consume when queuing records for streaming. The amount of space is limited to the value of this key-value pair multiplied by the maximum number of bytes per batch. The maximum bytes per batch for the `KinesisStream`, `KinesisFirehose`, and `CloudWatchLogs` sink types are 5 MB, 4 MB, and 1 MB respectively. When this limit is reached, any streaming failures are not queued and are reported as non-recoverable failures. This key-value pair is optional. The default value is 10,000 batches.

Configuring a Proxy for Sinks

To configure a proxy for all the Kinesis Agent for Windows sink types that access AWS services, edit the Kinesis Agent for Windows configuration file located at `%Program Files%\Amazon \KinesisTap\AWSKinesisTap.exe.config`. For instructions, see the `proxy` section in Configuration Files Reference for AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

Configuring resolving variables in more sink attributes

The following example shows a sink configuration that uses the `Region` environment variable for the value of the `Region` attribute key-value pair. For `RoleARN`, it specifies the EC2 tag key `MyRoleARN`, which evaluates to the value associated with that key.

```
"Id": "myCloudWatchLogsSink",
"SinkType": "CloudWatchLogs",
"LogGroup": "EC2Logs",
"LogStream": "logs-{instance_id}"
"Region": "{env:Region}"
"RoleARN": "{ec2tag:MyRoleARN}"
```

Configuring AWS STS Regional Endpoints When Using RoleARN Property in AWS Sinks

This feature only applies if you are using KinesisTap on Amazon EC2 and using the `RoleARN` property of AWS sinks to assume an external IAM role to authenticate with the destination AWS services.

By setting `UseSTSRegionalEndpoints` to `true`, you can specify that an agent use the regional endpoint (for example, `https://sts.us-east-1.amazonaws.com`) instead of the global endpoint (for example, `https://sts.amazonaws.com`). Using a Regional STS endpoint reduces round-trip latency for the operation and limits the impact of failures in the global endpoint service.

Configuring VPC Endpoint for AWS Sinks

You can specify a VPC endpoint in the sink configuration for `CloudWatchLogs`, `CloudWatch`, `KinesisStreams`, and `KinesisFirehose` sink types. A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network. For more information, see VPC endpoints in the *Amazon VPC User Guide*.

You specify the VPC endpoint using the `ServiceURL` property as shown in the following example of a `CloudWatchLogs` sink configuration. Set the value of `ServiceURL` to the value shown on the **VPC endpoint details** tab using the Amazon VPC console.

```
{
    "Id": "myCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "LogGroup": "EC2Logs",
    "LogStream": "logs-{instance_id}",
    "ServiceURL":"https://vpce-ab1c234de56-ab7cdefg.logs.us-east-1.vpce.amazonaws.com"
}
```

Configuring An Alternate Means of Proxy

This feature allows you to configure a proxy server in a sink configuration using the proxy support built in to the AWS SDK instead of .NET. Previously, the only way to configure the agent to use a

proxy was to use a native feature of .NET, which automatically routed all HTTP/S requests through the proxy defined in the proxy file.

If you are currently using the agent with a proxy server, you do not need to change over to use this method.

You can use the `ProxyHost` and `ProxyPort` properties to configure an alternate proxy as shown in the following example.

```
{
    "Id": "myCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "LogGroup": "EC2Logs",
    "LogStream": "logs-{instance_id}",
    "Region": "us-west-2",
    "ProxyHost": "myproxy.mydnsdomain.com",
    "ProxyPort": "8080"
}
```

Pipe Declarations

Use *pipe declarations* to connect a source (see Source Declarations) to a sink (see Sink Declarations) in Amazon Kinesis Agent for Microsoft Windows. A pipe declaration is expressed as a JSON object. After Kinesis Agent for Windows starts, the logs, events, or metrics are gathered from the source for a given pipe. They are then streamed to various AWS services using the sink that is associated with that pipe.

The following is an example pipe declaration:

```
{
   "Id": "MyAppLogToCloudWatchLogs",
   "SourceRef": "MyAppLog",
   "SinkRef": "MyCloudWatchLogsSink"
}
```

Topics

- Configuring Pipes
- Configuring Kinesis Agent for Windows Metric Pipes

Configuring Pipes

All pipe declarations can contain the following key-value pairs:

`Id`

Specifies the name of the pipe (required). It must be unique within the configuration file.

`Type`

Specifies the type of transformation (if any) that is applied by the pipe as log data is transferred from the source to the sink. The only supported value is `RegexFilterPipe`. This value enables regular expression filtering of the underlying textual representation of the log record. Using filtering can reduce transmission and storage costs by sending only relevant log records downstream to the data pipeline. This key-value pair is optional. The default value is to provide no transformation.

`FilterPattern`

Specifies the regular expression for `RegexFilterPipe` pipelines that are used to filter log records gathered by the source before being transferred to the sink. Log records are transferred by `RegexFilterPipe` type pipes when the regular expression matches the underlying textual representation of the record. Structured log records that are generated, for example, when using the `ExtractionPattern` key-value pair in a `DirectorySource` declaration, can still be filtered using the `RegexFilterPipe` mechanism. This is because this mechanism operates against the original textual representation before parsing. This key-value pair is optional, but it must be provided if the pipe specifies the `RegexFilterPipe` type.

The following is an example `RegexFilterPipe` pipe declaration:

```
{
 "Id": "MyAppLog2ToFirehose",
 "Type": "RegexFilterPipe",
 "SourceRef": "MyAppLog2",
 "SinkRef": "MyFirehose",
 "FilterPattern": "^(10|11),.*",
 "IgnoreCase": false,
 "Negate": false
}
```

`SourceRef`

Specifies the name (the value of the `Id` key-value pair) of the source declaration that defines the source that is collecting log, event, and metric data for the pipe (required).

`SinkRef`

Specifies the name (the value of the `Id` key-value pair) of the sink declaration that defines the sink that is receiving the log, event, and metric data for the pipe (required).

`IgnoreCase`

Optional. Accepts values of `true` or `false`. When set to `true`, the Regex will match records in a case-insensitive manner.

`Negate`

Optional. Accepts values of `true` or `false`. When set to `true`, the pipe will forward the records that *do not* match the regular expression.

For an example of a complete configuration file that uses the `RegexFilterPipe` pipe type, see Using Pipes.

Configuring Kinesis Agent for Windows Metric Pipes

There is a built-in metric source named `_KinesisTapMetricsSource` that produces metrics about Kinesis Agent for Windows. If there is a `CloudWatch` sink declaration with an `Id` of `MyCloudWatchSink`, the following example pipeline declaration transfers Kinesis Agent for Windows-generated metrics to that sink:

```
{
   "Id": "KinesisAgentMetricsToCloudWatch",
   "SourceRef": "_KinesisTapMetricsSource",
   "SinkRef": "MyCloudWatchSink"
}
```

For more information about the Kinesis Agent for Windows built-in metrics source, see Kinesis Agent for Windows Built-In Metrics Source.

If the configuration file also streams Windows performance counter metrics, we recommend that you use a separate pipe and sink rather than using the same sink for both Kinesis Agent for Windows metrics and Windows performance counter metrics.

Configuring Automatic Updates

Use the `appsettings.json` configuration file to enable automatic updating of Amazon Kinesis Agent for Microsoft Windows and the configuration file for Kinesis Agent for Windows. To control the update behavior, specify the `Plugins` key-value pair at the same level in the configuration file as `Sources`, `Sinks`, and `Pipes`.

The `Plugins` key-value pair specifies the additional general functionality to use that does not fall specifically under the categories of sources, sinks, and pipes. For example, there is a plugin for updating Kinesis Agent for Windows, and there is a plugin for updating the `appsettings.json` configuration file. Plugins are represented as JSON objects and always have a `Type` key-value pair. The `Type` determines what other key-value pairs can be specified for the plugin. The following plugin types are supported:

`PackageUpdate`

Specifies that Kinesis Agent for Windows should periodically check a package version configuration file. If the package version file indicates that a different version of Kinesis Agent for Windows should be installed, then Kinesis Agent for Windows downloads that version and installs it. The `PackageUpdate` plugin key-value pairs include:

`Type`

The value must be the string `PackageUpdate`, and it is required.

`Interval`

Specifies how often to check the package version file for any changes in minutes represented as a string. This key-value pair is optional. If it is not specified, the default value is 60 minutes. If the value is less than 1, no update checking occurs.

`PackageVersion`

Specifies the location of the package version JSON file. The file can reside on a file share (`file://`), a website (`http://`), or Amazon S3 (`s3://`). For example, a value of `s3://mycompany/config/agent-package-version.json` indicates that Kinesis Agent for Windows should check the contents of the `config/agent-package-version.json` file in the `mycompany` Amazon S3 bucket. It should perform updates based on the contents of that file.

> **Note**
>
> The value of the `PackageVersion` key-value pair is case sensitive for Amazon S3.

The following is an example of the contents of a package version file:

```
{
    "Name": "AWSKinesisTap",
    "Version": "1.0.0.106",
    "PackageUrl": "https://s3-us-west-2.amazonaws.com/kinesis-agent-windows/
downloads/AWSKinesisTap.{Version}.nupkg"
}
```

The `Version` key-value pair specifies what version of Kinesis Agent for Windows should be installed if it is not already installed. The `{Version}` variable reference in the `PackageUrl` resolves the value you specify for the `Version` key-value pair. In this example, the variable resolves to the string `1.0.0.106`. This variable resolution is provided so that there can be a single place in the package version file where the specific desired version is stored. You can use multiple package version files to control the pace of rolling out new versions of Kinesis Agent for Windows to validate a new version before a larger deployment. To roll back a deployment of Kinesis Agent for Windows, change one or more package version files to specify an earlier version of Kinesis Agent for Windows that is known to work in your environment.

The value of the `PackageVersion` key-value pair is affected by variable substitution to facilitate the automatic selection of different package version files. For more information about variable substitution, see Configuring Sink Variable Substitutions.

`AccessKey`

Specifies which access key to use when authenticating access to the package version file in Amazon S3. This key-value pair is optional. We do not recommend using this key-value pair. For alternative authentication approaches that are recommended, see Configuring Authentication.

`SecretKey`

Specifies which secret key to use when authenticating access to the package version file in Amazon S3. This key-value pair is optional. We do not recommend using this key-value

pair. For alternative authentication approaches that are recommended, see Configuring Authentication.

`Region`

Specifies what Region endpoint to use when accessing the package version file from Amazon S3. This key-value pair is optional.

`ProfileName`

Specifies which security profile to use when authenticating access to the package version file in Amazon S3. For more information, see Configuring Authentication. This key-value pair is optional.

`RoleARN`

Specifies which role to assume when authenticating access to the package version file in Amazon S3 in a cross-account scenario. For more information, see Configuring Authentication. This key-value pair is optional.

If no `PackageUpdate` plugin is specified, then no package version files are checked to determine if an update is required.

`ConfigUpdate`

Specifies that Kinesis Agent for Windows should periodically check for an updated `appsettings.json` configuration file stored in a file share, website, or Amazon S3. If an updated configuration file exists, it is downloaded and installed by Kinesis Agent for Windows. `ConfigUpdate` key-value pairs include the following:

`Type`

The value must be the string `ConfigUpdate`, and it is required.

`Interval`

Specifies how often to check for a new configuration file in minutes represented as a string. This key-value pair is optional, and if not specified, defaults to 5 minutes. If the value is less than 1, then the configuration file update is not checked.

`Source`

Specifies where to look for an updated configuration file. The file can reside on a file share (`file://`), a website (`http://`), or Amazon S3 (`s3://`). For example, a value of `s3://mycompany/config/appsettings.json` indicates that Kinesis Agent for Windows should

check for updates to the `config/appsettings.json` file in the `mycompany` Amazon S3 bucket.

> **Note**
>
> The value of the `Source` key-value pair is case-sensitive for Amazon S3.

The value of the `Source` key-value pair is affected by variable substitution to facilitate the automatic selection of different configuration files. For more information about variable substitution, see Configuring Sink Variable Substitutions.

`Destination`

Specifies where to store the configuration file on the local machine. This can be a relative path, an absolute path, or a path containing environment variable references such as `%PROGRAMDATA%`. If the path is relative, it is relative to the location where Kinesis Agent for Windows is installed. Typically the value should be `.\appsettings.json`. This key-value pair is required.

`AccessKey`

Specifies which access key to use when authenticating access to the configuration file in Amazon S3. This key-value pair is optional. We do not recommend using this key-value pair. For alternative authentication approaches that are recommended, see Configuring Authentication.

`SecretKey`

Specifies which secret key to use when authenticating access to the configuration file in Amazon S3. This key-value pair is optional. We do not recommend using this key-value pair. For alternative authentication approaches that are recommended, see Configuring Authentication.

`Region`

Specifies what Region endpoint to use when accessing the configuration file from Amazon S3. This key-value pair is optional.

`ProfileName`

Specifies which security profile to use when authenticating access to the configuration file in Amazon S3. For more information, see Configuring Authentication. This key-value pair is optional.

`RoleARN`

Specifies which role to assume when authenticating access to the configuration file in Amazon S3 in a cross-account scenario. For more information, see Configuring Authentication. This key-value pair is optional.

If no `ConfigUpdate` plugin is specified, then no configuration files are checked to determine whether a configuration file update is required.

The following is an example `appsettings.json` configuration file that demonstrates using the `PackageUpdate` and `ConfigUpdate` plugins. In this example, there is package version file located in the `mycompany` Amazon S3 bucket named `config/agent-package-version.json`. This file is checked for any changes approximately every 2 hours. If a different version of Kinesis Agent for Windows is specified in the package version file, the specified agent version is installed from the specified location in the package version file.

In addition, there is an `appsettings.json` configuration file stored in the `mycompany` Amazon S3 bucket named `config/appsettings.json`. Approximately every 30 minutes, that file is compared against the current configuration file. If they are different, the updated configuration file is downloaded from Amazon S3 and installed to the typical local location for the `appsettings.json` configuration file.

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
```

```
        "SinkType": "KinesisFirehose",
        "StreamName": "ApplicationLogFirehoseDeliveryStream",
        "Region": "us-east-1"
      }
      ],
  "Pipes": [
    {
      "Id": "ApplicationLogSourceToApplicationLogKinesisFirehoseSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink"
    }
  ],
  "Plugins": [
    {
      "Type": "PackageUpdate"
      "Interval": "120",
      "PackageVersion": "s3://mycompany/config/agent-package-version.json"
    },
    {
      "Type": "ConfigUpdate",
      "Interval": "30",
      "Source": "s3://mycompany/config/appsettings.json",
      "Destination": ".\appSettings.json"
    }
  ]
}
```

Kinesis Agent for Windows Configuration Examples

The `appsettings.json` configuration file is a JSON document that controls how Amazon Kinesis Agent for Microsoft Windows collects logs, events, and metrics. It also controls how Kinesis Agent for Windows transforms that data and streams it to various AWS services. For details about the source, sink, and pipe declarations in the configuration file, see Source Declarations, Sink Declarations, and Pipe Declarations.

The following sections contain examples of configuration files for several different kinds of scenarios.

Topics

- Streaming from Various Sources to Kinesis Data Streams
- Streaming from the Windows Application Event Log to Sinks

- Using Pipes
- Using Multiple Sources and Pipes

Streaming from Various Sources to Kinesis Data Streams

The following example `appsettings.json` configuration files demonstrate streaming logs and events from various sources to Kinesis Data Streams and from Windows performance counters to Amazon CloudWatch metrics.

`DirectorySource`, `SysLog` Record Parser

The following file streams syslog format log records from all files with a `.log` file extension in the `C:\LogSource\` directory to the `SyslogKinesisDataStream` Kinesis Data Streams stream in the us-east-1 Region. A bookmark is established to ensure that all data from the log files is sent even if the agent is shut down and restarted later. A custom application can read and process the records from the `SyslogKinesisDataStream` stream.

```
{
  "Sources": [
    {
      "Id": "SyslogDirectorySource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SysLog",
      "TimeZoneKind": "UTC",
      "InitialPosition": "Bookmark"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "SyslogKinesisDataStream",
      "Region": "us-east-1"
    }
  ],
  "Pipes": [
    {
      "Id": "SyslogDS2KSSink",
      "SourceRef": "SyslogDirectorySource",
```

```
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

`DirectorySource`, `SingleLineJson` Record Parser

The following file streams JSON-formatted log records from all files with a `.log` file extension in the `C:\LogSource\` directory to the `JsonKinesisDataStream` Kinesis Data Streams stream in the us-east-1 Region. Before streaming, key-value pairs for the `ComputerName` and `DT` keys are added to each JSON object, with values for the computer name and the date and time the record is processed. A custom application can read and process the records from the `JsonKinesisDataStream` stream.

```
{
  "Sources": [
    {
      "Id": "JsonLogSource",
      "SourceType": "DirectorySource",
      "RecordParser": "SingleLineJson",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "InitialPosition": 0
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "JsonKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "JsonLogSourceToKinesisStreamSink",
      "SourceRef": "JsonLogSource",
      "SinkRef": "KinesisStreamSink"
    }
```

```
  ]
}
```

ExchangeLogSource

The following file streams log records generated by Microsoft Exchange and stored in files with the `.log` extension in the `C:\temp\ExchangeLog\` directory to the `ExchangeKinesisDataStream` Kinesis data stream in the us-east-1 Region in JSON format. Although the Exchange logs are not in JSON format, Kinesis Agent for Windows can parse the logs and transform them to JSON. Before streaming, key-value pairs for the `ComputerName` and `DT` keys are added to each JSON object containing values for the computer name and the date and time the record is processed. A custom application can read and process the records from the `ExchangeKinesisDataStream` stream.

```
{
  "Sources": [
    {
       "Id": "ExchangeSource",
       "SourceType": "ExchangeLogSource",
       "Directory": "C:\\temp\\ExchangeLog\",
       "FileNameFilter": "*.log"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "ExchangeKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "ExchangeSourceToKinesisStreamSink",
      "SourceRef": "ExchangeSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

W3SVCLogSource

The following file streams Internet Information Services (IIS) for Windows log records stored in the standard location for those files to the IISKinesisDataStream Kinesis Data Streams stream in the us-east-1 Region. A custom application can read and process the records from the IISKinesisDataStream stream. IIS is a web server for Windows.

```
{
  "Sources": [
    {
       "Id": "IISLogSource",
       "SourceType": "W3SVCLogSource",
       "Directory": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
       "FileNameFilter": "*.log"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "IISKinesisDataStream",
      "Region": "us-east-1"
    }
  ],
  "Pipes": [
    {
      "Id": "IISLogSourceToKinesisStreamSink",
      "SourceRef": "IISLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

WindowsEventLogSource with Query

The following file streams log events from the Windows system event log that have a level of Critical or Error (less than or equal to 2) to the SystemKinesisDataStream Kinesis data stream in the us-east-1 Region in JSON format. A custom application can read and process the records from the SystemKinesisDataStream stream.

```
{
  "Sources": [
    {
         "Id": "SystemLogSource",
         "SourceType": "WindowsEventLogSource",
         "LogName": "System",
         "Query": "*[System/Level<=2]"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "SystemKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "SLSourceToKSSink",
      "SourceRef": "SystemLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

`WindowsETWEventSource`

The following file streams Microsoft Common Language Runtime (CLR) exception and security events to the `ClrKinesisDataStream` Kinesis data stream in the us-east-1 Region in JSON format. A custom application can read and process the records from the `ClrKinesisDataStream` stream.

```
{
  "Sources": [
    {
       "Id": "ClrETWEventSource",
       "SourceType": "WindowsETWEventSource",
       "ProviderName": "Microsoft-Windows-DotNETRuntime",
       "TraceLevel": "Verbose",
       "MatchAnyKeyword": "0x00008000, 0x00000400"
```

```
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "ClrKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "ETWSourceToKSSink",
      "SourceRef": "ClrETWEventSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

`WindowsPerformanceCounterSource`

The following file streams performance counters for total files open, total login attempts since reboot, number of disk reads per second, and percentage of free disk space to CloudWatch metrics in the us-east-1 Region. You can graph these metrics in CloudWatch, build dashboards from the graphs, and set alarms that send notifications when thresholds are exceeded.

```
{
  "Sources": [
    {
      "Id": "PerformanceCounter",
      "SourceType": "WindowsPerformanceCounterSource",
      "Categories": [
        {
          "Category": "Server",
          "Counters": [
            "Files Open",
            "Logon Total"
          ]
        },
        {
          "Category": "LogicalDisk",
          "Instances": "*",
```

```
          "Counters": [
            "% Free Space",
            {
              "Counter": "Disk Reads/sec",
              "Unit": "Count/Second"
            }
          ]
        }
      ],
    }
  ],
  "Sinks": [
    {
      "Namespace": "MyServiceMetrics",
      "Region": "us-east-1",
      "Id": "CloudWatchSink",
      "SinkType": "CloudWatch"
    }
  ],
  "Pipes": [
    {
      "Id": "PerformanceCounterToCloudWatch",
      "SourceRef": "PerformanceCounter",
      "SinkRef": "CloudWatchSink"
    }
  ]
}
```

Streaming from the Windows Application Event Log to Sinks

The following example `appsettings.json` configuration files demonstrate streaming Windows application event logs to various sinks in Amazon Kinesis Agent for Microsoft Windows. For examples of using the `KinesisStream` and `CloudWatch` sink types, see Streaming from Various Sources to Kinesis Data Streams.

`KinesisFirehose`

The following file streams `Critical` or `Error` Windows application log events to the `WindowsLogFirehoseDeliveryStream` Firehose delivery stream in the us-east-1 Region. If connectivity to Firehose is interrupted, events are first queued in memory. Then if necessary, they are queued to a file on disk until connectivity is restored. Then events are unqueued and sent followed by any new events.

You can configure Firehose to store the streamed data to several different kinds of storage and analysis services based on data pipeline requirements.

```
{
  "Sources": [
    {
         "Id": "ApplicationLogSource",
         "SourceType": "WindowsEventLogSource",
         "LogName": "Application",
         "Query": "*[System/Level<=2]"
    }
  ],
  "Sinks": [
    {
       "Id": "WindowsLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "WindowsLogFirehoseDeliveryStream",
       "Region": "us-east-1",
       "QueueType": "file"
    }
    ],
  "Pipes": [
    {
      "Id": "ALSource2ALKFSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "WindowsLogKinesisFirehoseSink"
    }
  ]
}
```

`CloudWatchLogs`

The following file streams `Critical` or `Error` Windows application log events to CloudWatch Logs log streams in the `MyServiceApplicationLog-Group` log group. The name of each stream begins with `Stream-`. It ends with the four-digit year, two-digit month, and two-digit day that the stream was created, all concatenated (for example, `Stream-20180501` is the stream created on May 1, 2018).

```
{
  "Sources": [
    {
         "Id": "ApplicationLogSource",
```

```
        "SourceType": "WindowsEventLogSource",
        "LogName": "Application",
        "Query": "*[System/Level<=2]"
    }
  ],
  "Sinks": [
    {
      "Id": "CloudWatchLogsSink",
      "SinkType": "CloudWatchLogs",
      "LogGroup": "MyServiceApplicationLog-Group",
      "LogStream": "Stream-{timestamp:yyyyMMdd}",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "ALSource2CWLSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "CloudWatchLogsSink"
    }
  ]
}
```

Using Pipes

The following example `appsettings.json` configuration file demonstrates using pipe-related features.

This example streams log entries from the `c:\LogSource\` to the `ApplicationLogFirehoseDeliveryStream` Firehose delivery stream. It includes only lines that match the regular expression specified by the `FilterPattern` key-value pair. Specifically, only lines in the log file that start with `10` or `11` are streamed to Firehose.

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
```

```
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ALSourceToALKFSink",
      "Type": "RegexFilterPipe",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink",
      "FilterPattern": "^(10|11),.*"
    }
  ]
}
```

Using Multiple Sources and Pipes

The following example `appsettings.json` configuration file demonstrates using multiple sources and pipes.

This example streams the application, security, and system Windows Event Logs to the `EventLogStream` Firehose delivery stream using three sources, three pipes, and a single sink.

```
{
    "Sources": [
  {
    "Id": "ApplicationLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Application"
  },
  {
    "Id": "SecurityLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Security"
  },
  {
    "Id": "SystemLog",
```

```
    "SourceType": "WindowsEventLogSource",
    "LogName": "System"
  }
    ],
    "Sinks": [
  {
    "Id": "EventLogSink",
    "SinkType": "KinesisFirehose",
    "StreamName": "EventLogStream",
    "Format": "json"
  },
    ],
    "Pipes": [
  {
    "Id": "ApplicationLogToFirehose",
    "SourceRef": "ApplicationLog",
    "SinkRef": "EventLogSink"
  },
  {
    "Id": "SecurityLogToFirehose",
    "SourceRef": "SecurityLog",
    "SinkRef": "EventLogSink"
  },
  {
    "Id": "SystemLogToFirehose",
    "SourceRef": "SystemLog",
    "SinkRef": "EventLogSink"
  }
    ]
}
```

Configuring Telemetrics

To enable better support, by default, Amazon Kinesis Agent for Microsoft Windows collects statistics about the operation of the agent and sends them to AWS. This information contains no personally identifiable information. It doesn't include any data that you gather or stream to AWS services. We collect approximately 1–2 KB of this metric data every 60 minutes.

You can opt out of the collection and transmission of these statistics. To do this, add the following key-value pair to the `appsettings.json` configuration file at the same level as sources, sinks, and pipes:

```
"Telemetrics":
    { "off": "true" }
```

For example, the following configuration file configures a source, sink, and pipe, and also disables telemetrics:

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ApplicationLogSourceToApplicationLogKinesisFirehoseSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink"
    }
  ],
  "Telemetrics":
    {
      "off": "true"
    }
}
```

We collect the following metrics when telemetry is enabled:

`ClientId`

The automatically assigned unique ID when the software is installed.

`ClientTimestamp`

The date and time the telemetry is collected.

`OSDescription`

A description of the operating system.

`DotnetFramework`

The current dotnet framework version.

`MemoryUsage`

The amount of memory consumed by Kinesis Agent for Windows(MB).

`CPUUsage`

The amount of Kinesis Agent for Windows CPU usage percentage in decimal. For example, 0.01 means 1%.

`InstanceId`

The Amazon EC2 instance ID if Kinesis Agent for Windows is running on an Amazon EC2 instance.

`InstanceType (string)`

The Amazon EC2 instance type if Kinesis Agent for Windows is running on an Amazon EC2 instance.

In addition, we collect the metrics listed in List of Kinesis Agent for Windows Metrics.

Tutorial: Stream JSON Log Files to Amazon S3 Using Kinesis Agent for Windows

This tutorial presents detailed steps for setting up a data pipeline using Amazon Kinesis Agent for Microsoft Windows (Kinesis Agent for Windows).

The tutorial includes the following steps:

- Using Kinesis Agent for Windows to stream JSON-formatted log files to Amazon Simple Storage Service (Amazon S3) via Amazon Data Firehose. For information about Kinesis Agent for Windows, see What Is Amazon Kinesis Agent for Microsoft Windows?.
- Enhancing the log data before streaming using object decoration. For more information, see Configuring Sink Decorations.
- Using Amazon Athena to search for particular kinds of log records.

Prerequisites

If you don't already have an AWS account, follow the instructions in Setting Up an AWS account to get one.

Topics

Step 1: Configure AWS services

Follow these steps to prepare your environment for streaming log data to Amazon Simple Storage Service (Amazon S3) using Amazon Kinesis Agent for Microsoft Windows. For more information and prerequisites, see *Tutorial: Stream JSON Log Files to Amazon S3*.

Use the AWS Management Console to configure AWS Identity and Access Management (IAM), Amazon S3, Firehose, and Amazon Elastic Compute Cloud (Amazon EC2) to prepare for streaming log data from an EC2 instance to Amazon S3.

Topics

Configure IAM Policies and Roles

Create the following policy, which authorizes Kinesis Agent for Windows to stream records to a specific Firehose delivery stream:

Replace the example Region, *us-east-1* with the name of the AWS Region where the Firehose delivery stream will be created. Also, replace the example account, *123456789012* with the 12-digit account ID for the AWS account where the delivery stream will be created.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:us-
east-1:123456789012:deliverystream/log-delivery-stream"
        }
    ]
}
```

In the navigation bar, choose **Support**, and then **Support Center**. Your currently signed-in 12-digit account number (ID) appears in the **Support Center** navigation pane.

Create the policy using the following procedure. Name the policy `log-delivery-stream-access-policy`.

To create a policy using the JSON policy editor

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane on the left side, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.
3. At the top of the page, choose **Create policy**.
4. Choose the **JSON** tab.
5. Enter a JSON policy document. For details about the IAM policy language, see IAM JSON Policy Reference in the *IAM User Guide*.
6. When you are finished, choose **Review policy**. The Policy Validator reports any syntax errors.

   > **Note**
   >
   > You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy Restructuring in the *IAM User Guide*.
7. On the **Review policy** page, enter a **Name** and a **Description** (optional) for the policy that you are creating. Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy** to save your work.

To create the role that gives Firehose access to an S3 bucket

1. Using the previous procedure, create a policy named `firehose-s3-access-policy` that is defined using the following JSON.

   Replace the following in your IAM policy example:

   - The example Amazon S3 bucket name, *`amzn-s3-demo-bucket`* with a unique bucket name where the logs will be stored.

- The example Region, *us-east-1* with the AWS Region where the CloudWatch Logs log group and log stream will be created. These are for logging any errors that occur during streaming the data to Amazon S3 via Firehose.
- The example AWS account ID, *123456789012* with the 12-digit account ID for the account where the log group and log stream will be created.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:us-east-1:123456789012:log-group:firehose-error-
log-group:log-stream:firehose-error-log-stream"
           ]
        }
    ]
}
```

2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS service** role type, and then choose the **Kinesis** service.
4. Choose **Firehose** for the use case, and then choose **Next: Permissions**.
5. In the search box, enter **`firehose-s3-access-policy`**, choose that policy, and then choose **Next: Review**.
6. In the **Role name** box, enter **`firehose-s3-access-role`**.
7. Choose **Create role**.

To create the role to associate with the instance profile for the EC2 instance that will run Kinesis Agent for Windows

1. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
2. Choose the **AWS service** role type, and then choose **EC2**.
3. Choose **Next: Permissions**.
4. In the search box, enter **`log-delivery-stream-access-policy`**.
5. Choose the policy, and then choose **Next: Review**.
6. In the **Role name** box, enter **`kinesis-agent-instance-role`**.
7. Choose **Create role**.

Create the Amazon S3 Bucket

Create the S3 bucket where Firehose streams the logs.

To create the S3 bucket for log storage

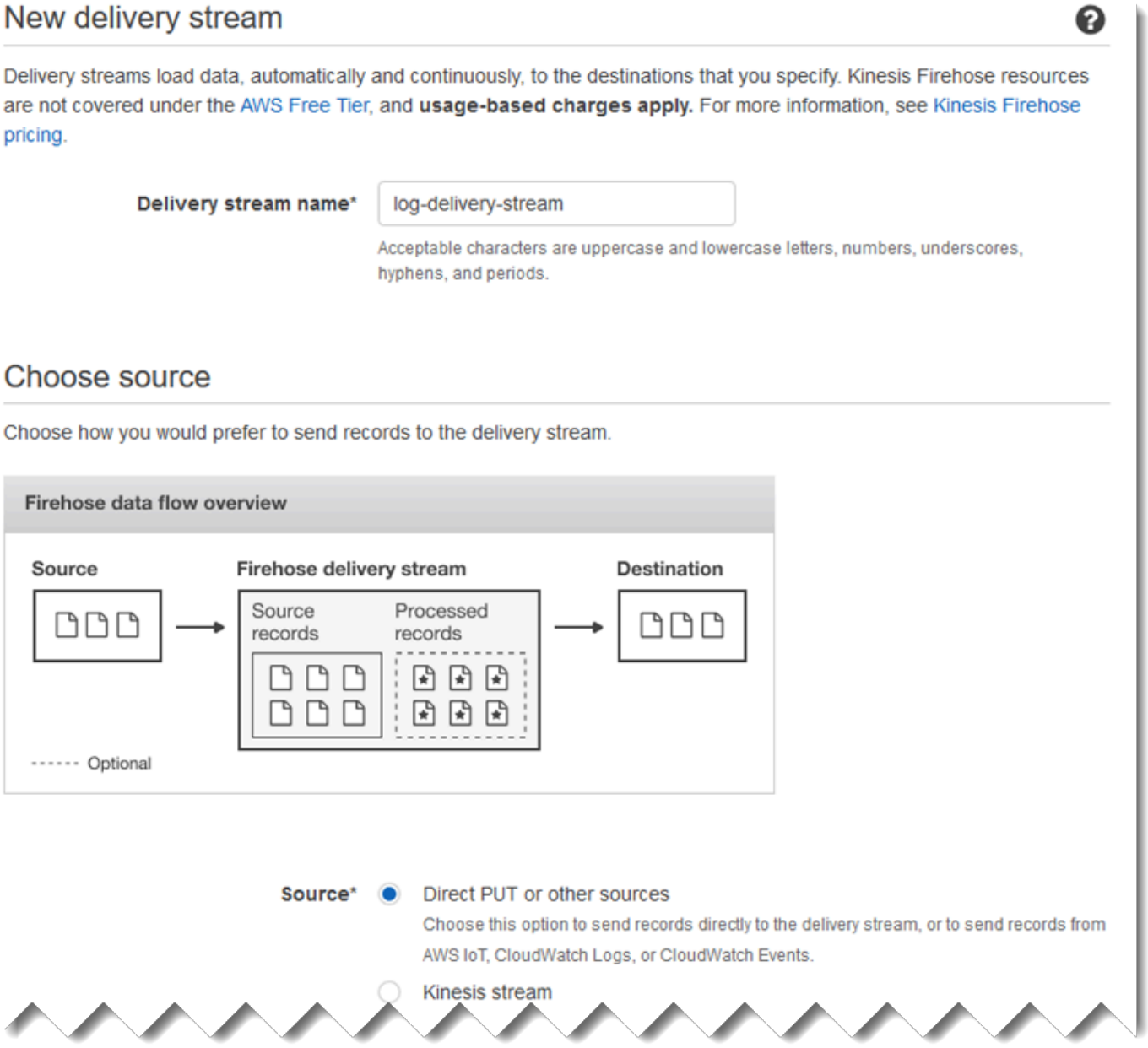
1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create bucket**.
3. In the **Bucket name** box, enter the unique S3 bucket name that you chose in Configure IAM Policies and Roles.
4. Choose the Region where the bucket should be created. This is typically the same Region where you intend to create the Firehose delivery stream and the Amazon EC2 instance.
5. Choose **Create**.

Create the Firehose Delivery Stream

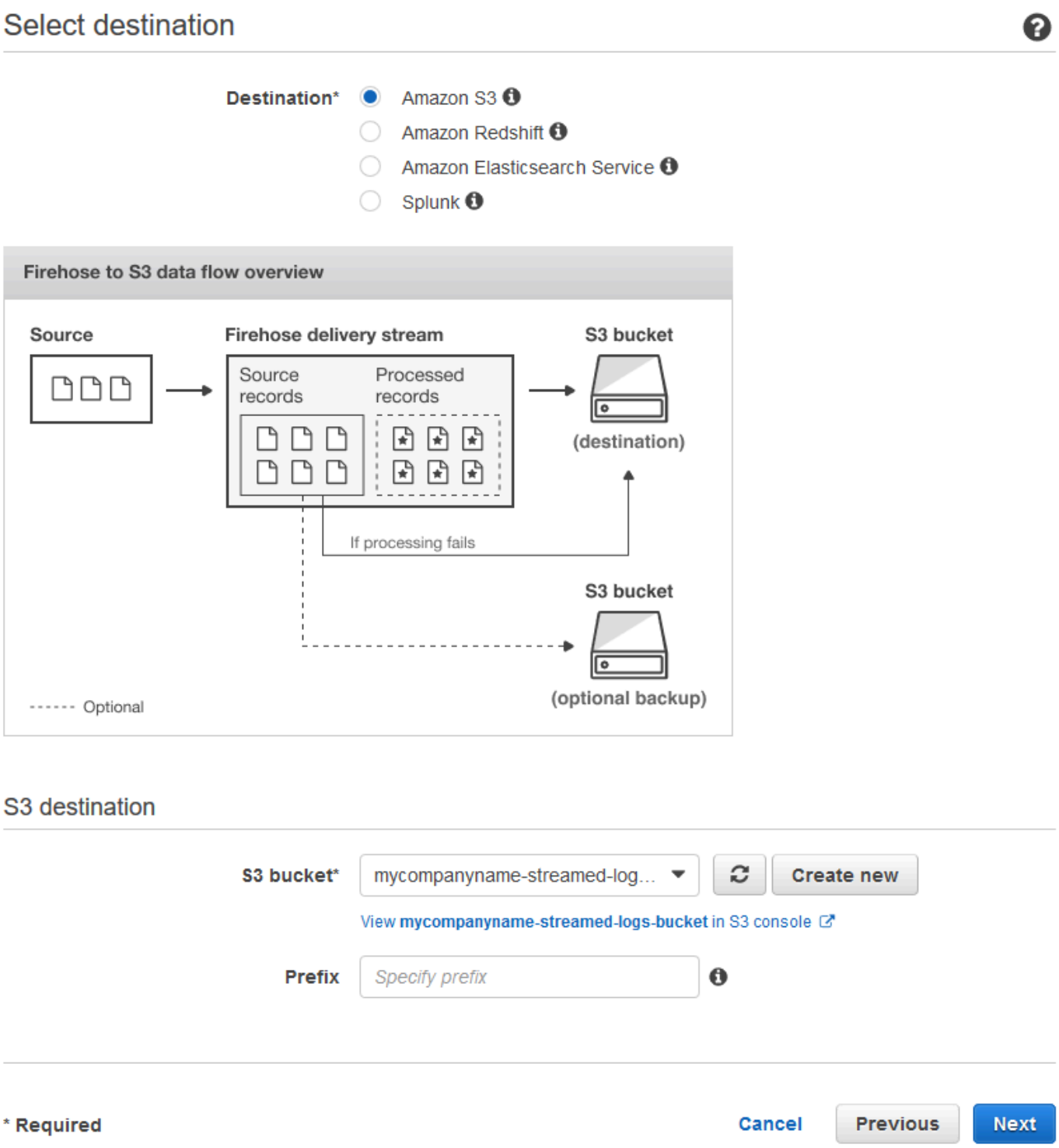
Create the Firehose delivery stream that will store streamed records in Amazon S3.

To create the Firehose delivery stream

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. Choose **Create Delivery Stream**.
3. In the **Delivery stream name** box, enter **`log-delivery-stream`**.
4. For the **Source**, choose **Direct PUT or other sources**.



5. Choose **Next**.
6. Choose **Next** again.
7. For the destination, choose **Amazon S3**.
8. For the **S3 bucket**, choose the name of the bucket that you created in Create the Amazon S3 Bucket.



9. Choose **Next**.
10. In the **Buffer interval** box, enter **60**.
11. Under **IAM role**, choose **Create new or choose**.
12. For **IAM role**, choose `firehose-s3-access-role`.
13. Choose **Allow**.

Configure settings

Configure buffer, compression, logging, and IAM role settings for your delivery stream.

S3 buffer conditions

Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery will be triggered once either of these conditions has been satisfied. Learn more

Buffer size* 5 MB

Specify a buffer size between 1-128 MB

Buffer interval* 60 seconds

Specify a buffer interval between 60-900 seconds

S3 compression and encryption

Firehose can compress records before delivering them to your S3 bucket. Compressed records can also be encrypted in the S3 bucket using a KMS master key. Learn more

S3 compression* (•) Disabled ( ) GZIP ( ) Snappy ( ) Zip

S3 encryption* (•) Disabled ( ) Enabled

Error logging

Firehose can log record delivery errors to CloudWatch Logs. If enabled, a CloudWatch log group and corresponding log streams are created on your behalf. Learn more

Error logging* ( ) Disabled (•) Enabled

IAM role

Firehose uses an IAM role to access your specified resources, such as the S3 bucket and KMS key. Learn more

IAM role* firehose-s3-access-role

Create new or choose

14. Choose **Next**.
15. Choose **Create delivery stream**.

Create the Amazon EC2 Instance to Run Kinesis Agent for Windows

Create the EC2 instance that uses Kinesis Agent for Windows to stream log records via Firehose.

To create the EC2 instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Follow the instructions in Getting Started with Amazon EC2 Windows Instances, using the following additional steps:
   - For the **IAM role** for the instance, choose `kinesis-agent-instance-role`.
   - If you don't already have a public internet-connected virtual private cloud (VPC), follow the instructions in Setting Up with Amazon EC2 in the *Amazon EC2 User Guide*.
   - Create or use a security group that limits access to the instance from only your computer, or only your organization's computers. For more information, see Setting Up with Amazon EC2 in the *Amazon EC2 User Guide*.
   - If you specify an existing key pair, be sure to have access to the private key for the key pair. Or, create a new key pair and save the private key in a safe place.
   - Before continuing, wait until the instance is running and has completed two out of two health checks.
   - Your instance requires a public IP address. If one hasn't been allocated, follow the instructions at Elastic IP Addresses in the *Amazon EC2 User Guide*.

Next Steps

Step 2: Install, Configure, and Run Kinesis Agent for Windows

Step 2: Install, Configure, and Run Kinesis Agent for Windows

In this step, you use the AWS Management Console to remotely connect to the instance that you launched in Create the Amazon EC2 Instance to Run Kinesis Agent for Windows. You then install Amazon Kinesis Agent for Microsoft Windows on the instance, create and deploy the configuration file for Kinesis Agent for Windows, and start the **`AWSKinesisTap`** service.

1. Remotely connect to the instance via Remore Desktop Protocol (RDP) by following the instructions in Step 2: Connect to Your Instance in the *Amazon EC2 User Guide*.
2. On the instance, use Windows Server Manager to disable Microsoft Internet Explorer Enhanced Security Configuration for users and administrators. For more information, see How To Turn Off Internet Explorer Enhanced Security Configuration on the Microsoft TechNet website.
3. On the instance, install and configure Kinesis Agent for Windows. For more information, see Installing Kinesis Agent for Windows.
4. On the instance, use Notepad to create a Kinesis Agent for Windows configuration file. Save the file to `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json`. Add the following content to the configuration file:

```
{
  "Sources": [
    {
      "Id": "JsonLogSource",
      "SourceType": "DirectorySource",
      "RecordParser": "SingleLineJson",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "InitialPosition": 0
    }
  ],
  "Sinks": [
    {
      "Id": "FirehoseLogStream",
      "SinkType": "KinesisFirehose",
      "StreamName": "log-delivery-stream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "JsonLogSourceToFirehoseLogStream",
      "SourceRef": "JsonLogSource",
      "SinkRef": "FirehoseLogStream"
    }
  ]
```

```
}
```

This file configures Kinesis Agent for Windows to send JSON-formatted log records from files in the `c:\logsource\` directory (the *source*) to a Firehose delivery stream named `log-delivery-stream` (the *sink*). Before each log record is streamed to Firehose, it is enhanced with two extra key-value pairs that contain the name of the computer and a timestamp.

5. Create the `c:\LogSource\` directory, and use Notepad to create a `test.log` file in that directory with the following content:

```
{ "Message": "Copasetic message 1", "Severity": "Information" }
{ "Message": "Copasetic message 2", "Severity": "Information" }
{ "Message": "Problem message 2", "Severity": "Error" }
{ "Message": "Copasetic message 3", "Severity": "Information" }
```

6. In an elevated PowerShell session, use the following command to start the **`AWSKinesisTap`** service:

```
Start-Service -ServiceName AWSKinesisTap
```

7. Using File Explorer, browse to the `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` directory. Open the most recent log file. The log file should look similar to the following:

```
2018-09-28 23:51:02.2472 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AWS.AWSEventSinkFactory.
2018-09-28 23:51:02.2784 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.PerformanceCounterSinkFactory.
2018-09-28 23:51:02.5753 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.DirectorySourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.ExchangeSource.ExchangeSourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Uls.UlsSourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.WindowsSourceFactory.
2018-09-28 23:51:02.9347 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.Pipes.PipeFactory.
2018-09-28 23:51:03.5128 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AutoUpdate.AutoUpdateFactory.
2018-09-28 23:51:03.5440 Amazon.KinesisTap.Hosting.LogManager INFO Performance
 counter sink  started.
```

```
2018-09-28 23:51:03.7628 Amazon.KinesisTap.Hosting.LogManager INFO
 KinesisFirehoseSink id FirehoseLogStream for StreamName log-delivery-stream
 started.
2018-09-28 23:51:03.7784 Amazon.KinesisTap.Hosting.LogManager INFO Connected source
 JsonLogSource to sink FirehoseLogStream
2018-09-28 23:51:03.7940 Amazon.KinesisTap.Hosting.LogManager INFO DirectorySource
 id JsonLogSource watching directory C:\LogSource\ with filter *.log started.
```

This log file indicates that the service has started and log records are now being collected from the `c:\LogSource\` directory. Each line is parsed as a single JSON object. Key-value pairs for the computer name and timestamp are added to each object. Then it is streamed to Firehose.

8. In a minute or two, navigate to the Amazon S3 bucket that you created in Create the Amazon S3 Bucket using the AWS Management Console. Be sure that you have chosen the correct Region on the console.

   In that bucket, there is a folder for the current year. Open that folder to reveal a folder for the current month. Open that folder to reveal a folder for the current day. Open that folder to reveal a folder for the current hour (in UTC). Open that folder to reveal one or more items that start with the name `log-delivery-stream`.

9. Open the contents of the latest item to confirm that the log records have been successfully stored in Amazon S3 with the desired enhancements. If everything is configured correctly, the contents look similar to the following:

```
{"Message":"Copasetic message 1","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Copasetic message 2","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Problem message 2","Severity":"Error","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
```

```
{"Message":"Copasetic message 3","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
```

10. For information about resolving any of the following issues, see Troubleshooting Amazon Kinesis Agent for Microsoft Windows:

    - The Kinesis Agent for Windows log file contains errors.
    - Expected folders or items in Amazon S3 do not exist.
    - The contents of an Amazon S3 item are incorrect.

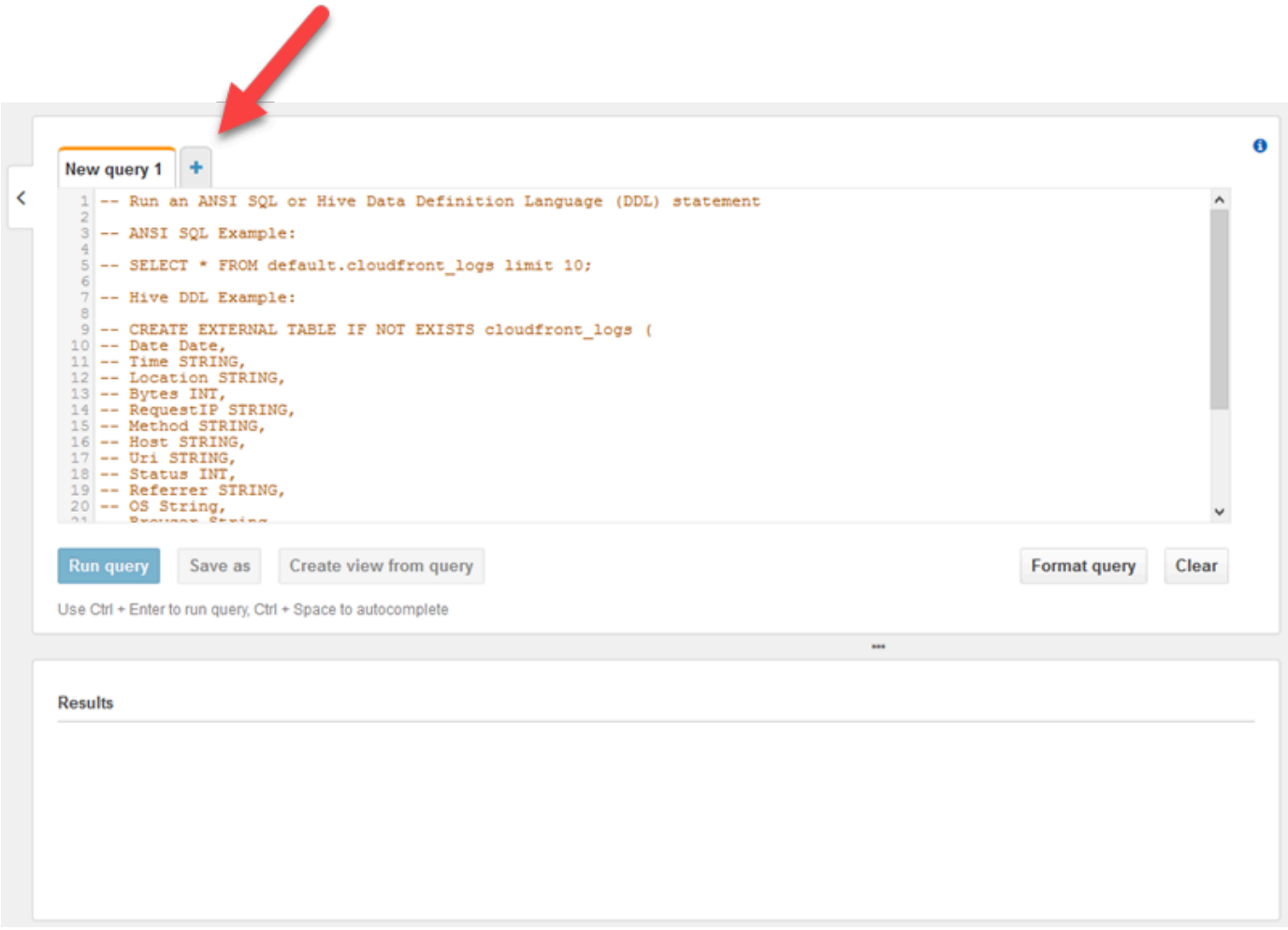
Next Steps

Step 3: Query the Log Data in Amazon S3

Step 3: Query the Log Data in Amazon S3

In the final step of this Amazon Kinesis Agent for Microsoft Windows tutorial, you use Amazon Athena to query the log data stored in Amazon Simple Storage Service (Amazon S3).

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. Choose the plus sign (**+**) in the Athena query window to create a new query window.

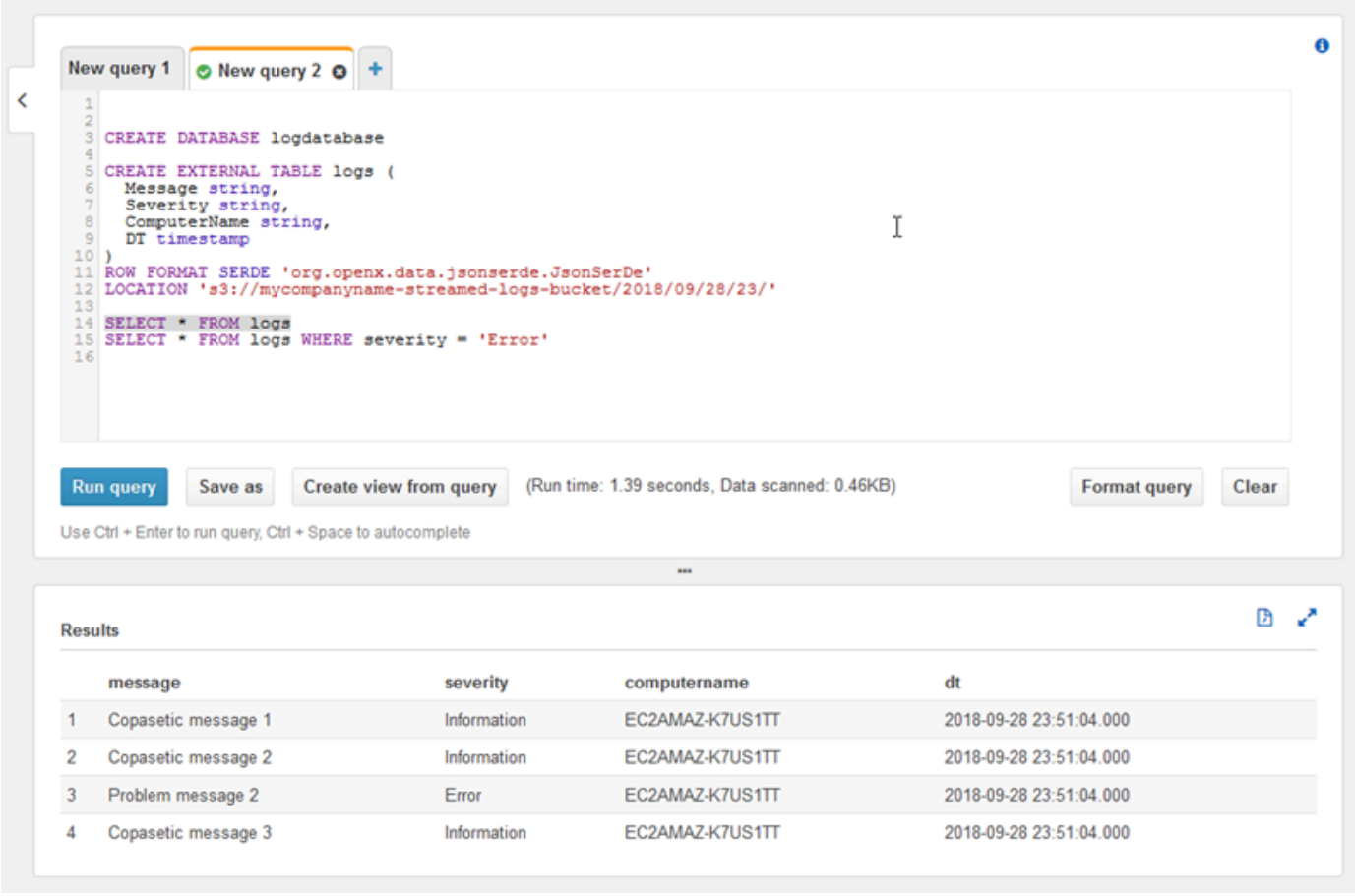


3. Enter the following text into the query window:

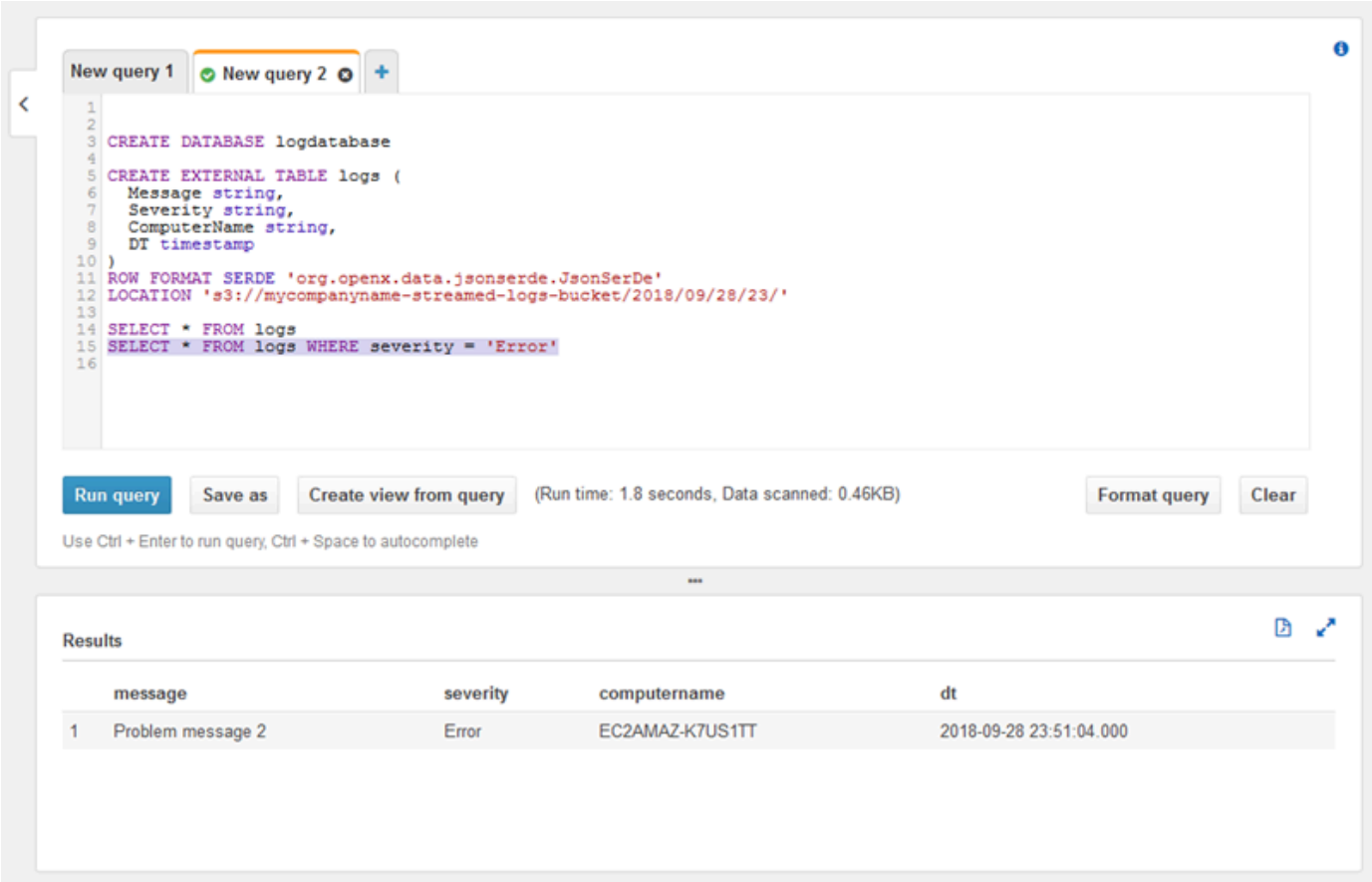
```
CREATE DATABASE logdatabase

CREATE EXTERNAL TABLE logs (
  Message string,
  Severity string,
  ComputerName string,
  DT timestamp
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://bucket/year/month/day/hour/'

SELECT * FROM logs
SELECT * FROM logs WHERE severity = 'Error'
```

Replace *bucket* with the name of the bucket that you created in Create the Amazon S3 Bucket. Replace *year*, *month*, *day* and *hour* with the year, month, day, and hour when the Amazon S3 log file was created in UTC.

4. Select the text for the CREATE DATABASE statement, and then choose **Run query**. This creates the log database in Athena.
5. Select the text for the CREATE EXTERNAL TABLE statement, and then choose **Run query**. This creates an Athena table that references the S3 bucket with the log data, mapping the schema for the JSON to the schema for the Athena table.
6. Select the text for the first SELECT statement, and then choose **Run query**. This displays all the rows in the table.



7. Select the text for the second SELECT statement, and then choose **Run query**. This displays only the rows in the table that represent log records with an `Error`-level severity. This kind of query finds interesting log records from a potentially large set of log records.



Next Steps

Use the AWS Management Console to clean up the resources created during the tutorial:

1. Terminate the EC2 instance (see step 3 in Getting Started with Amazon EC2 Windows Instances).

   > **Important**
   >
   > If you launched an instance that was not within the AWS Free Tier, you are charged for the instance until you terminate it.

2. Delete the Firehose delivery stream.

   a. Open the Firehose console at https://console.aws.amazon.com/firehose/.

   b. Choose the delivery stream that you created.

   c. Choose **Delete**.

d. Choose **Delete delivery stream**.

3. Delete the S3 bucket. For instructions, see How Do I Delete an S3 Bucket? in the *Amazon Simple Storage Service User Guide*.

For more information, see the following topics:

- Configuring Amazon Kinesis Agent for Microsoft Windows
- What Is Amazon Kinesis Data Firehose?
- What Is Amazon S3?
- What is Amazon Athena?

Troubleshooting Amazon Kinesis Agent for Microsoft Windows

Use the following instructions to diagnose and correct issues when using Amazon Kinesis Agent for Microsoft Windows.

Topics

No Data Is Streaming from Desktops or Servers to Expected AWS services

Symptoms

When you examine logs, events, and metrics hosted by various AWS services that are configured to receive streams of data from Kinesis Agent for Windows, no data is being streamed by Kinesis Agent for Windows.

Causes

There are several possible causes for this issue:

- A source, sink, or pipe is configured incorrectly.
- Authentication for Kinesis Agent for Windows is configured incorrectly.
- Authorization for Kinesis Agent for Windows is configured incorrectly.
- There is an error in a regular expression provided in a `DirectorySource` declaration.
- A nonexistent directory is specified for a `DirectorySource` declaration.

- Invalid values are being provided to AWS services, which then reject requests from Kinesis Agent for Windows.
- A sink is referencing a resource that doesn't exist in the specified or implicit AWS Region.
- An invalid query is specified for a `WindowsEventLogSource` declaration.
- An invalid value is specified for the `InitialPosition` key-value pair for a source.
- The `appsettings.json` configuration file does not comply with the JSON schema for that file.
- The data is streaming to a different Region than what is selected in the AWS Management Console.
- Kinesis Agent for Windows is not installed correctly or is not running.

Resolutions

To resolve issues with data not streaming, perform the following steps:

1. Examine the Kinesis Agent for Windows logs in the `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` directory. Search for the string `ERROR`.
   a. If a source or sink did not load, do the following:
      i. Examine the error message, and find the `Id` of the source or sink.
      ii. Check the source or sink declaration that corresponds to that `Id` in the `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` configuration file for any errors related to the error message found. For more details, see Configuring Amazon Kinesis Agent for Microsoft Windows.
      iii. Correct any configuration file issues related to the error.
      iv. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the configuration issues have been resolved.
   b. If the error message indicates that a `SourceRef` or `SinkRef` was not found for a pipe, do the following:
      i. Note the pipe `Id`.
      ii. Examine the pipe declaration in the `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` configuration file that corresponds to the noted `Id`. Ensure that the values of the `SourceRef` and `SinkRef` key-value pairs are correctly spelled `Ids` for the source and sink declarations that you intended to reference. Correct any typos or spelling errors. If a source or sink declaration is missing from the configuration file, add the

declaration. For more information, see Configuring Amazon Kinesis Agent for Microsoft Windows.

iii. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the configuration issues have been resolved.

c. If the error message indicates that a particular IAM user or role is not authorized to perform certain operations, do the following:

i. Ensure that the correct IAM user or role is being used by Kinesis Agent for Windows. If it is not, review Sink Security Configuration, and adjust how Kinesis Agent for Windows authenticates to ensure that the correct IAM user or role is being used.

ii. If the correct IAM user or role is being used, using the AWS Management Console, examine the policies that are associated with the user or role. Ensure that the user or role has all the permissions mentioned in the error message for all the AWS resources that Kinesis Agent for Windows accesses. For more information, see Configuring Authorization.

iii. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the security issues have been resolved.

d. If the error message indicates that there is an argument error when parsing a regular expression that is contained in the `%PROGRAMFILES%\Amazon\AWSKinesisTap \appsettings.json` configuration file, do the following:

i. Examine the regular expression in the configuration file.

ii. Verify the syntax of the regular expression. There are several websites that you can use to verify regular expressions, or use the following command lines to check regular expressions for a `DirectorySource` source declaration:

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag.exe /r sourceId
```

Replace *`sourceId`* with the value of the `Id` key-value pair of the `DirectorySource` source declaration with an incorrect regular expression.

iii. Make any corrections necessary to the regular expression in the configuration file so that it is valid.

iv. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the configuration issues have been resolved.

e. If the error message indicates that there is an argument error when parsing a regular expression that is not contained in the `%PROGRAMFILES%\Amazon\AWSKinesisTap`

\appsettings.json configuration file, and that is related to a particular sink, do the following:

i. Locate the sink declaration in the configuration file.

ii. Verify that the key-value pairs that are specifically related to an AWS service are using names that comply with the validation rules for that service. For example, CloudWatch Logs group names must contain only a certain set of characters that is specified using the regular expression [\.\-_/#A-Za-z0-9]+.

iii. Correct any invalid names in the key-value pairs for the sink declaration, and ensure that those resources are properly configured in AWS.

iv. Stop and start the AWSKinesisTap service. Then check the most current log file to verify that the configuration issues have been resolved.

f. If the error message indicates that a source or sink cannot load due to a null or missing parameter, do the following:

i. Note the Id of the source or sink.

ii. Locate the source or sink declaration that matches the noted Id in the %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json configuration file.

iii. Review the key-value pairs that are provided in the source or sink declaration compared with the source or sink type requirements in the Configuring Amazon Kinesis Agent for Microsoft Windows documentation for the relevant sink type. Add any missing required key-value pairs to the source or sink declaration.

iv. Stop and start the AWSKinesisTap service. Then check the most current log file to verify that the configuration issues have been resolved.

g. If the error message indicates that a directory name is invalid, do the following:

i. Locate the invalid directory name in the %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json configuration file.

ii. Verify that this directory exists and contains the log files that should be streamed.

iii. Correct any typos or mistakes in the directory name specified in the configuration file.

iv. Stop and start the AWSKinesisTap service. Then check the most current log file to verify that the configuration issues have been resolved.

h. If the error message indicates that a resource does not exist:

i. Locate the resource reference for the resource that doesn't exist in a sink declaration in the %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json configuration file.

ii. Use the AWS Management Console to locate the resource in the correct AWS Region that should be used in the sink declaration. Compare it to what was specified in the configuration file.

iii. Change the sink declaration in the configuration file to have the correct resource name and the correct Region.

iv. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the configuration issues have been resolved.

i. If the error message indicates that a query is invalid for a particular `WindowsEventLogSource`, do the following:

i. In the `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` configuration file, locate the `WindowsEventLogSource` declaration with the same `Id` as in the error message.

ii. Verify that the value of the `Query` key-value pair in the source declaration complies with Event queries and Event XML.

iii. Make any changes to the query to bring it into compliance.

iv. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the configuration issues have been resolved.

j. If the error message indicates that there is an invalid initial position, do the following:

i. In the `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` configuration file, locate the source declaration with the same `Id` as the error message.

ii. Change the value of the `InitialPosition` key-value pair in the source declaration to comply with the permitted values, as described in Bookmark Configuration.

iii. Stop and start the `AWSKinesisTap` service. Then check the most current log file to verify that the configuration issues have been resolved.

2. Ensure that the `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` configuration file complies with the JSON schema.

a. In a command prompt window, invoke the following lines:

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
%PROGRAMFILES%\Amazon\AWSKinesisTap\ktdiag.exe /c
```

b. Correct any issues detected with the `%PROGRAMFILES%\Amazon\AWSKinesisTap \appsettings.json` configuration file.

c. Stop and start the AWSKinesisTap service. Then check the most current log file to verify that the configuration issues have been resolved.

3. Change the logging level to try to get more detailed logging information.

a. Replace the %PROGRAMFILES%\Amazon\AWSKinesisTap\nlog.xml configuration file with the following content:

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.nlog-project.org/schemas/NLog.xsd NLog.xsd"
      autoReload="true"
      throwExceptions="false"
      internalLogLevel="Off" internalLogFile="c:\temp\nlog-internal.log" >

  <!--
  See https://github.com/nlog/nlog/wiki/Configuration-file
  for information on customizing logging rules and outputs.
   -->
  <targets>
    <!--
    add your targets here
    See https://github.com/nlog/NLog/wiki/Targets for possible targets.
    See https://github.com/nlog/NLog/wiki/Layout-Renderers for the possible layout
 renderers.
    -->

    <target name="logfile"
            xsi:type="File"
            layout="${longdate} ${logger} ${uppercase:${level}} ${message}"
            fileName="${specialfolder:folder=CommonApplicationData}/Amazon/
KinesisTap/logs/KinesisTap.log"
     maxArchiveFiles="90"
     archiveFileName="${specialfolder:folder=CommonApplicationData}/Amazon/
KinesisTap/logs/Archive-{################}.log"
     archiveNumbering="Date"
     archiveDateFormat="yyyy-MM-dd"
     archiveEvery="Day"
     />
  </targets>

  <rules>
    <logger name="*" minlevel="Debug" writeTo="logfile" />
```

```
  </rules>
</nlog>
```

b. Stop and start the `AWSKinesisTap` service. Then check the most current log file to see if there are additional messages in the log that could help diagnose and resolve the issue.

4. Verify that you are looking at resources in the correct Region in the AWS Management Console.
5. Verify that the Kinesis Agent for Windows agent is installed and running.
   a. In Windows, choose **Start**, and then navigate to **Control Panel**, **Administrative Tools**, **Services**.
   b. Find the **`AWSKinesisTap`** service.
   c. If the `AWSKinesisTap` service is not visible, install Kinesis Agent for Windows using the instructions in Getting Started with Amazon Kinesis Agent for Microsoft Windows.
   d. If the service is visible, determine whether the service is running. If it is not running, open the context (right-click) menu for the service, and choose **Start**.
   e. Verify that the service has started by examining the latest log file in the `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` directory.

Applies to

This information applies to Kinesis Agent for Windows version 1.0.0.115 and higher.

Expected Data Is Sometimes Missing

Symptoms

Kinesis Agent for Windows streams data successfully most of the time, but occasionally some data is missing.

Causes

There are several possible causes for this issue:

- The bookmarking feature is not being used.
- Data rate limits for AWS services are exceeded based on the current configuration of those services.

- API call rates limits for AWS services are exceeded based on the current `appsettings.json` configuration file and the AWS account limits.

Resolutions

To resolve issues with missing data, perform the following steps:

1. Consider using the bookmarking feature documented in Bookmark Configuration. It helps ensure that all data is eventually sent, even when Kinesis Agent for Windows stops and starts.
2. Use Kinesis Agent for Windows's built-in metrics to discover problems:
   a. Enable the streaming of Kinesis Agent for Windows metrics as described in Configuring Kinesis Agent for Windows Metric Pipes.
   b. If there are a significant number of non-recoverable errors for one or more sinks, determine how many bytes or records are being sent per second. Then determine whether this is within the limits configured for those AWS services in the Region and account where the data is being streamed.
   c. When limits are exceeded, either reduce the rate or amount of data being sent, request limit increases, or increase sharding if applicable.
   d. After making adjustments, continue to monitor the Kinesis Agent for Windows built-in metrics to ensure that the situation has been resolved.

For more information on Kinesis Data Streams limits, see Kinesis Data Streams Limits in the *Kinesis Data Streams Developer Guide*. For more information on Firehose limits, see Amazon Kinesis Data Firehose Limits.

Applies to

This information applies to Kinesis Agent for Windows version 1.0.0.115 and higher.

Data Arrives in an Incorrect Format

Symptoms

Data arrives at the AWS service in the incorrect format.

Causes

There are several possible causes for this issue:

- The value for the `Format` key-value pair for a sink declaration in the `appsettings.json` configuration file is incorrect.
- The value for the `RecordParser` key-value pair in a `DirectorySource` declaration is incorrect.
- The regular expressions in a `DirectorySource` declaration that uses the `Regex` record parser are incorrect.

Resolutions

To resolve issues with incorrect formatting, perform the following steps:

1. Review the sink declarations in the `%PROGRAMFILES%\Amazon\AWSKinesisTap \appsettings.json` configuration file.
2. Ensure that the correct value of the `Format` key-value pair is specified for each sink declaration. For more information, see Sink Declarations.
3. If sources with `DirectorySource` declarations are connected by pipes to sinks that specify `xml` or `json` values for the `Format` key-value pair, ensure that those sources specify one of the following values for the `RecordParser` key-value pair:
   - `SingleLineJson`
   - `Regex`
   - `SysLog`
   - `Delimited`

   Other record parsers are text-based only and do not work correctly with sinks that require XML or JSON formatting.
4. If log records are not being correctly parsed by the `DirectorySource` source type, invoke the following lines in a command prompt window to verify the timestamp and regular expression key-value pairs specified in the `DirectorySource` declaration:

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag.exe /r sourceID
```

Replace *sourceID* with the value of the `Id` key-value pair of the `DirectorySource` source declaration that does not appear to be working correctly. Correct any problems reported by `ktdiag.exe`.

Applies to

This information applies to Kinesis Agent for Windows version 1.0.0.115 and higher.

Performance Issues

Symptoms

Applications and services have increased latencies after Kinesis Agent for Windows is installed and started.

Causes

There are several possible causes for this issue:

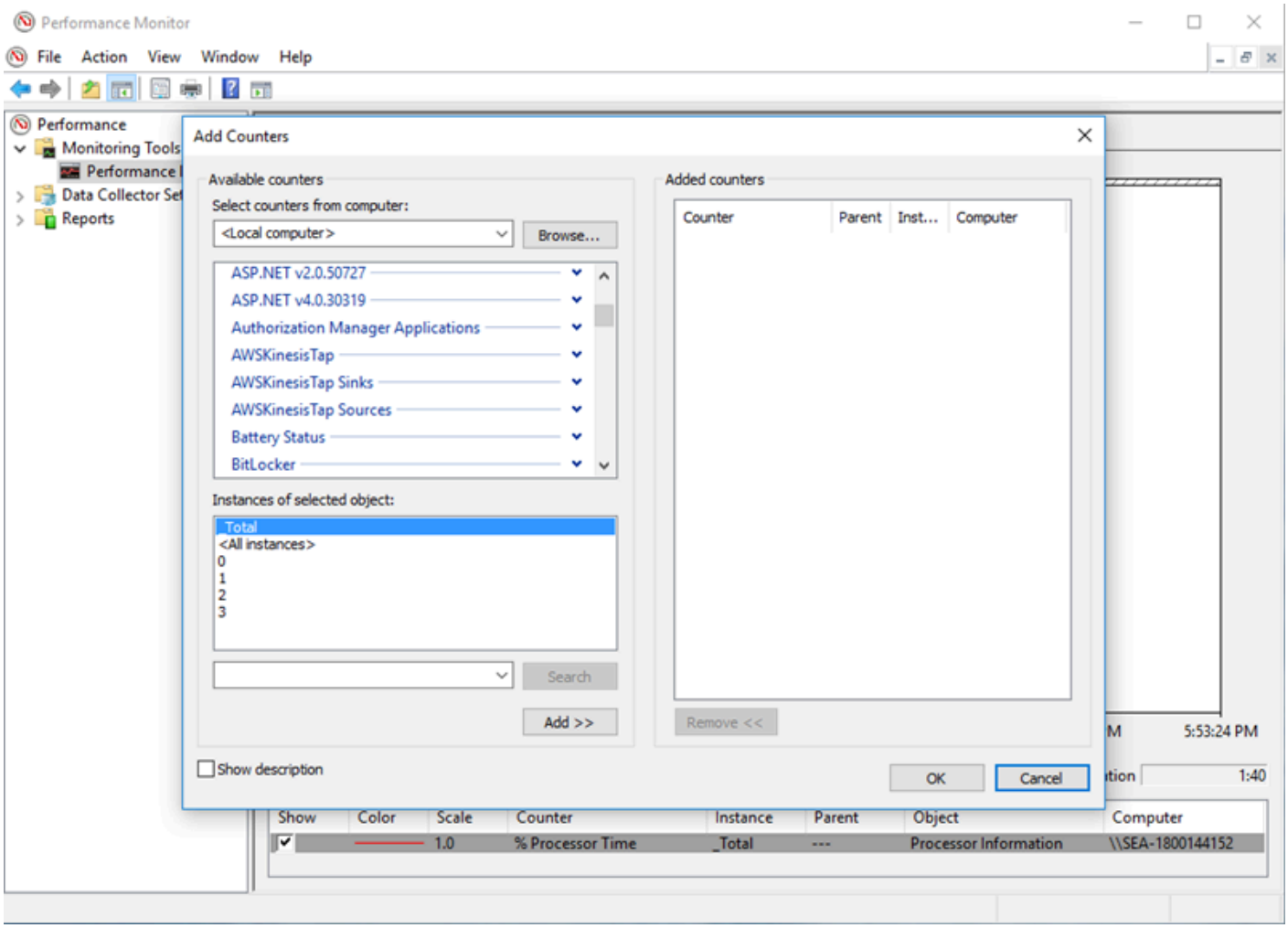
- The machine where Kinesis Agent for Windows runs does not have sufficient capacity to stream the amount of data desired.
- Unnecessary data is being streamed to one or more AWS services.
- Kinesis Agent for Windows is streaming data to AWS services that are not configured for such a high data rate.
- Kinesis Agent for Windows is invoking operations on AWS services in an account where the API call rate limit is too low.

Resolutions

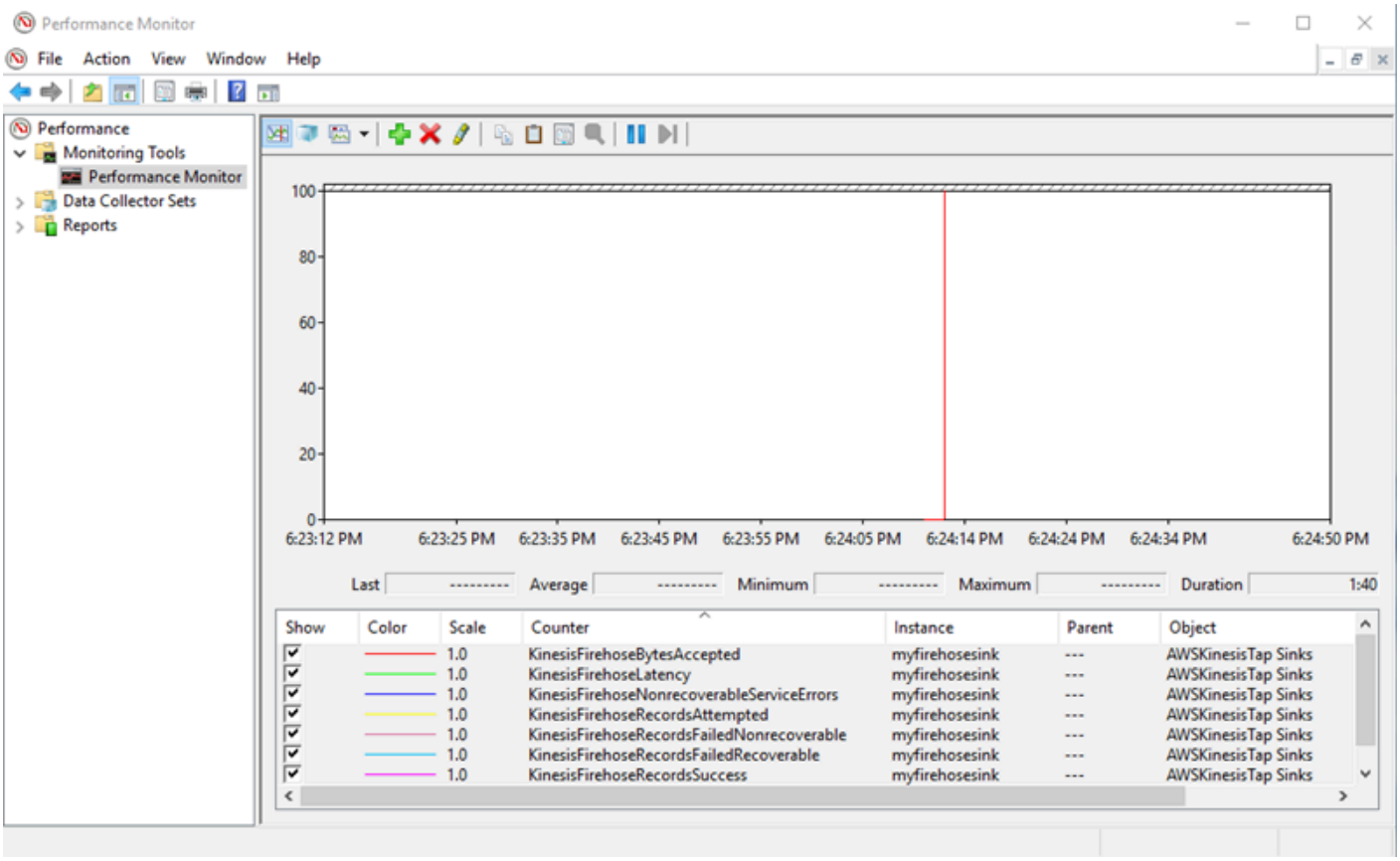
To resolve performance issues, perform the following steps:

1. Use the Windows resource monitor application to check memory, CPU, disk, and network usage. If you need to stream large quantities of data with Kinesis Agent for Windows, you might need to provision a machine with higher capacities in some of these areas, depending on configuration.

2. You might be able to reduce the amount of logged data using filtering:
   - See the `Query` key-value pair in WindowsEventLogSource Configuration.
   - See pipeline filtering in Configuring Pipes.
   - See Amazon CloudWatch metrics filtering in CloudWatch Sink Configuration).
3. Use the Windows performance monitor application to view Kinesis Agent for Windows metrics or stream those metrics to CloudWatch (see Kinesis Agent for Windows Built-In Metrics Source). In the Windows performance monitor application, you can add counters for Kinesis Agent for Windows sinks and sources. They are listed under the **`AWSKinesisTap Sinks`** and **`AWSKinesisTap Sources`** counter categories.



For example, to diagnose Firehose performance issues, add the **Kinesis Firehose Sink** performance counters.



If there are a large number of recoverable errors, inspect the latest Kinesis Agent for Windows logs in the `%PROGRAMDATA%\Amazon\AWSKInesisTap\logs` directory. If throttling is occurring for `KinesisStream` or `KinesisFirehose` sinks, do the following:

- If throttling occurs due to streaming data too quickly, consider raising the number of shards for the Kinesis data stream. For more information, see Resharding, Scaling, and Parallel Processing in the *Kinesis Data Streams Developer Guide*.
- Consider raising the API call limit for Kinesis Data Streams, or increasing the buffer size for the sink if the API calls are being throttled. For more information, see Kinesis Data Streams Limits in the *Kinesis Data Streams Developer Guide*.
- If data is streaming too quickly, consider requesting a rate limit increase for the Firehose delivery stream. Or if the API calls are being throttled, request an API call limit increase (see Amazon Kinesis Data Firehose Limits) or increase the buffer size for the sink.
- After increasing the number of shards for a Kinesis Data Streams stream, or increasing the rate limit for a Firehose delivery stream, revise the Kinesis Agent for Windows `appsettings.json` configuration file to increase the records per second or bytes per second for the sink. Otherwise, Kinesis Agent for Windows cannot take advantage of the increased limits.

Applies to

This information applies to Kinesis Agent for Windows version 1.0.0.115 and higher.

Out of Disk Space

Symptoms

Kinesis Agent for Windows is running on a machine that is very low on disk space on one or more disk drives.

Causes

There are several possible causes for this issue:

- The Kinesis Agent for Windows logging configuration file is incorrect.
- The Kinesis Agent for Windows persistent queue is configured incorrectly.
- Some other application or service is consuming disk space.

Resolutions

To resolve disk space issues, perform the following steps:

- If the disk space is low on the disk that contains the Kinesis Agent for Windows log files, examine the log file directory (typically `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs`). Ensure that a reasonable number of log files are being retained and that the log files are a reasonable size. You can control the location, retention, and verbosity of the Kinesis Agent for Windows logs by editing the `%PROGRAMFILES%\Amazon\AWSKinesisTap\Nlog.xml` configuration file.
- When the sink queuing feature is enabled, examine the sink declarations that use that feature. Ensure that the `QueuePath` key-value pair references a disk drive with sufficient space to contain the maximum number of batches specified using the `QueueMaxBatches` key-value pair. If this is not possible, then reduce the value of the `QueueMaxBatches` key-value pair so that the data easily fits in the remaining disk space for the specified disk drive.
- Use the Windows file explorer to locate the files consuming the disk space and either transfer or delete excess files. Change the configuration of the application or service consuming large amounts of disk space.

Applies to

This information applies to Kinesis Agent for Windows version 1.0.0.115 and higher.

Troubleshooting Tools

In addition to verifying the configuration file, you can use the `ktdiag.exe` tool, which provides several other capabilities for diagnosing and resolving problems when configuring and using Kinesis Agent for Windows. The `ktdiag.exe` tool is located in the `%PROGRAMFILES%\Amazon\AWSKinesisTap` directory.

- If you think that log files with a certain file pattern are being written to a directory but are not being processed by Kinesis Agent for Windows, use the `/w` switch to verify that these changes are being detected. For example, suppose that you expect that log files with the `*.log` file name pattern are being written to the `c:\foo` directory. You can use the `/w` switch when executing the `ktdiag.exe` tool, specifying the directory and file pattern:

  ```
  cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
  ktdiag /w c:\foo *.log
  ```

  If log files are being written, you can see output similar to the following:

  ```
  Type any key to exit this program...
  File: c:\foo\log1.log ChangeType: Created
  File: c:\foo\log1.log ChangeType: Deleted
  File: c:\foo\log1.log ChangeType: Created
  File: c:\foo\log1.log ChangeType: Changed
  File: c:\foo\log1.log ChangeType: Changed
  File: c:\foo\log1.log ChangeType: Changed
  File: c:\foo\log1.log ChangeType: Changed
  ```

  If no such output is occurring, then there is an application or service issue in writing the logs, or there is a security configuration issue rather than a problem with Kinesis Agent for Windows. If such output is occurring but Kinesis Agent for Windows is still not apparently processing the logs, see No Data Is Streaming from Desktops or Servers to Expected AWS services.
- Sometimes logs are only occasionally written, but it would be useful to verify that Kinesis Agent for Windows is operating correctly. Use the `/log4net` switch to simulate an application writing logs using the `Log4net` library; for example:

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
KTDiag.exe /log4net c:\foo\log2.log
```

This writes a `Log4net` style log file to the `c:\foo\log2.log` log file and keeps adding new log entries until a key is pressed. You can configure several options using additional switches that are optionally specified after the file name:

Locking: `-lm`, `-li` or `-le`

You can specify one of the following locking switches that control how the log file is locked:

`-lm`

The minimum amount of locking is used on the log file, enabling maximum access to the log file.

`-li`

Only threads within the same process can access the log at the same time.

`-le`

Only one thread at a time can access log. This is the default.

`-tn:`*`milliseconds`*

Specifies the number of *`milliseconds`* between writing log entries. The default is 1000 milliseconds (1 second).

`-sm:`*`bytes`*

Specifies the number of *`bytes`* for each log entry. The default is 1000 bytes.

`-bk:`*`number`*

Specifies the *`number`* of log entries to write at a time. The default is 1.

- Sometimes it is useful to simulate an application that writes to the Windows event log. Use the `/e` switch to write log entries a Windows event log; for example:

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
KTDiag.exe /e Application
```

This writes log entries to the Windows Application event log until a key is pressed. You can optionally specify the following additional options after the name of the log:

`-tn:`*`milliseconds`*

Specifies the number of *`milliseconds`* between writing log entries. The default is 1000 milliseconds (1 second).

`-sm:`*`bytes`*

Specifies the number of *`bytes`* for each log entry. The default is 1000 bytes.

`-bk:`*`number`*

Specifies the *`number`* of log entries to write at a time. The default is 1.

Creating Kinesis Agent for Windows Plugins

For most situations, creating an Amazon Kinesis Agent for Microsoft Windows plugin is not necessary. Kinesis Agent for Windows is highly configurable and contains powerful sources and sinks, such as `DirectorySource` and `KinesisStream`, which are sufficient for most scenarios. For details about the existing sources and sinks, see Configuring Amazon Kinesis Agent for Microsoft Windows.

For unusual scenarios, it might be necessary to extend Kinesis Agent for Windows using a custom plugin. Some of these scenarios include the following:

- Packaging a complex `DirectorySource` declaration using the `Regex` or `Delimited` record parsers so that it is easy to apply in many different kinds of configuration files.
- Creating a novel source that is not file based or that exceeds the parsing capabilities provided by the existing record parsers.
- Creating a sink for an AWS service that is not currently supported.

Topics

Getting Started with Kinesis Agent for Windows Plugins

There is nothing special about custom plugins. All the existing sources and sinks use the same mechanisms that custom plugins use to load when Kinesis Agent for Windows starts up, and they instantiate relevant plugins after reading the `appsettings.json` configuration file.

When Kinesis Agent for Windows starts up, the following sequence occurs:

1. Kinesis Agent for Windows scans assemblies in the installation directory (`%PROGRAMFILES%\Amazon\AWSKinesisTap`) for classes that implement the `IFactory<T>` interface defined in the `Amazon.KinesisTap.Core` assembly. This interface is defined in

`Amazon.KinesisTap.Core\Infrastructure\IFactory.cs` in the Kinesis Agent for Windows source code.

2. Kinesis Agent for Windows loads the assemblies containing these classes and invokes the `RegisterFactory` method on these classes.
3. Kinesis Agent for Windows loads the `appsettings.json` configuration file. For each source and sink in the configuration file, the `SourceType` and `SinkType` key-value pairs are examined. If there are factories registered with the same name as the values of the `SourceType` and `SinkType` key-value pairs, the `CreateInstance` method is invoked on those factories. The `CreateInstance` method is passed configuration and other information as an `IPluginContext` object. The `CreateInstance` method is responsible for configuring and initializing the plugin.

For a plugin to work correctly, there must be a registered factory class that creates the plugin, and the plugin class itself must be defined.

The Kinesis Agent for Windows source code is located at Kinesis Agent windows.

Implementing Kinesis Agent for Windows Plugin Factories

Follow these steps to implement a Kinesis Agent for Windows plugin factory.

To create a Kinesis Agent for Windows plugin factory

1. Create a C# library project targeting .NET Framework 4.6.
2. Add a reference to the `Amazon.KinesisTap.Core` assembly. This assembly is located in the `%PROGRAMFILES%\Amazon\AWSKinesisTap` directory after Kinesis Agent for Windows installation.
3. Use `NuGet` to install the `Microsoft.Extensions.Configuration.Abstractions` package.
4. Use `NuGet` to install the `System.Reactive` package.
5. Use `NuGet` to install the `Microsoft.Extensions.Logging` package.
6. Create a factory class that implements either `IFactory<IEventSource>` for sources or `IFactory<IEventSink>` for sinks. Add the `RegisterFactory` and `CreateInstance` methods.

For example, the following code creates a Kinesis Agent for Windows plugin factory that creates a source that generates random data:

```
using System;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Configuration;

namespace MyCompany.MySources
{
    public class RandomSourceFactory : IFactory<ISource>
    {
        public void RegisterFactory(IFactoryCatalog<ISource> catalog)
        {
            catalog.RegisterFactory("randomsource", this);
        }

        public ISource CreateInstance(string entry, IPlugInContext context)
        {
            IConfiguration config = context.Configuration;

            switch (entry.ToLower())
            {
                case "randomsource":
                    string rateString = config["Rate"];
                    string maxString = config["Max"];
                    TimeSpan rate;
                    int max;

                    if (string.IsNullOrWhiteSpace(rateString))
                    {
                        rate = TimeSpan.FromSeconds(30);
                    }
                    else
                    {
                        if (!TimeSpan.TryParse(rateString, out rate))
                        {
                            throw new Exception($"Rate {rateString} is invalid for
 RandomSource.");
                        }
                    }

                    if (string.IsNullOrWhiteSpace(maxString))
```

```
                    {
                        max = 1000;
                    }
                    else
                    {
                        if (!int.TryParse(maxString, out max))
                        {
                            throw new Exception($"Max {maxString} is invalid for
 RandomSource.");
                        }
                    }

                    return new RandomSource(rate, max, context);
                default:
                    throw new ArgumentException($"Source {entry} is not
 recognized.", entry);
            }
        }
    }
}
```

The `switch` statement is used in the `CreateInstance` method in case you eventually want to enhance the factory to create different kinds of instances.

To create a sink factory that creates a sink that does nothing, use a class similar to the following:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Configuration;

namespace MyCompany.MySinks
{
    public class NullSinkFactory : IFactory<IEventSink>
    {
        public void RegisterFactory(IFactoryCatalog<IEventSink> catalog)
        {
            catalog.RegisterFactory("nullsink", this);
        }
```

```
        public IEventSink CreateInstance(string entry, IPlugInContext context)
        {
            IConfiguration config = context.Configuration;

            switch (entry.ToLower())
            {
                case "nullsink":
                    return new NullSink(context);
                default:
                    throw new Exception("Unrecognized sink type {entry}.");
            }
        }
    }
}
```

Implementing Kinesis Agent for Windows Plugin Sources

Follow these steps to implement a Kinesis Agent for Windows plugin source.

To create a Kinesis Agent for Windows plugin source

1. Add a class that implements the `IEventSource<out T>` interface to the previously created project for the source.

   For example, use the following code to define a source that generates random data:

```
using System;
using System.Reactive.Subjects;
using System.Timers;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Logging;

namespace MyCompany.MySources
{
    public class RandomSource : EventSource<RandomData>, IDisposable
    {
        private TimeSpan _rate;
        private int _max;
        private Timer _timer = null;
        private Random _random = new Random();
```

```
        private ISubject<IEnvelope<RandomData>> _recordSubject = new
Subject<IEnvelope<RandomData>>();



        public RandomSource(TimeSpan rate, int max, IPlugInContext context) :
base(context)
        {
            _rate = rate;
            _max = max;
        }

        public override void Start()
        {
            try
            {
                CleanupTimer();
                _timer = new Timer(_rate.TotalMilliseconds);
                _timer.Elapsed += (Object source, ElapsedEventArgs args) =>
                {
                    var data = new RandomData()
                    {
                        RandomValue = _random.Next(_max)
                    };
                    _recordSubject.OnNext(new Envelope<RandomData>(data));
                };
                _timer.AutoReset = true;
                _timer.Enabled = true;
                _logger?.LogInformation($"Random source id {this.Id} started with
rate {_rate.TotalMilliseconds}.");
            }
            catch (Exception e)
            {
                _logger?.LogError($"Exception during start of RandomSource id
{this.Id}: {e}");
            }
        }

        public override void Stop()
        {
            try
            {
                CleanupTimer();
```

```
                _logger?.LogInformation($"Random source id {this.Id} stopped.");

            }
            catch (Exception e)
            {
                _logger?.LogError($"Exception during stop of RandomSource id
 {this.Id}: {e}");
            }
        }

        private void CleanupTimer()
        {
            if (_timer != null)
            {
                _timer.Enabled = false;
                _timer?.Dispose();
                _timer = null;
            }
        }

        public override IDisposable Subscribe(IObserver<IEnvelope<RandomData>>
 observer)
        {
            return this._recordSubject.Subscribe(observer);
        }

        public void Dispose()
        {
            CleanupTimer();
        }
    }
}
```

In this example, the `RandomSource` class inherits from the `EventSource<T>` class because it provides the `Id` property. Although this example doesn't support bookmarking, this base class is also useful for implementing that functionality. Envelopes provide a way to store metadata and wrap arbitrary data for streaming to sinks. The `RandomData` class is defined in the next step and represents the type of output object from this source.

2. Add a class to the previously defined project that contains the data that is streamed from the source.

   For example, a container for random data could be defined as the following:

```
namespace MyCompany.MySources
{
    public class RandomData
    {
        public int RandomValue { get; set; }
    }
}
```

3. Compile the previously defined project.
4. Copy the assembly to the installation directory for Kinesis Agent for Windows.
5. Create or update an `appsettings.json` configuration file that uses the new source, and place it in the installation directory for Kinesis Agent for Windows.
6. Stop and then start Kinesis Agent for Windows.
7. Check the current Kinesis Agent for Windows log file (usually located in the `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` directory) to ensure that there are no issues with the custom source plugin.
8. Ensure that data is arriving at the desired AWS service.

For an example of how to extend the `DirectorySource` functionality to implement parsing of a particular log format, see `Amazon.KinesisTap.Uls\UlsSourceFactory.cs` and `Amazon.KinesisTap.Uls\UlsLogParser.cs` in the Kinesis Agent for Windows source code.

For an example of how to create a source that provides bookmarking functionality, see `Amazon.KinesisTap.Windows\WindowsSourceFactory.cs` and `Amazon.KinesisTap.Windows\EventLogSource.cs` in the Kinesis Agent for Windows source code.

Implementing Kinesis Agent for Windows Plugin Sinks

Follow these steps to implement a Kinesis Agent for Windows plugin sink.

To create a Kinesis Agent for Windows plugin sink

1. Add a class to the previously defined project that implements the `IEventSink` interface.

   For example, the following code implements a sink that does nothing other than log the arrival of records, which are then discarded.

```
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Logging;

namespace MyCompany.MySinks
{
    public class NullSink : EventSink
    {
        public NullSink(IPlugInContext context) : base(context)
        {
        }

        public override void OnNext(IEnvelope envelope)
        {
            _logger.LogInformation($"Null sink {Id} received
 {GetRecord(envelope)}.");
        }

        public override void Start()
        {
            _logger.LogInformation($"Null sink {Id} starting.");
        }

        public override void Stop()
        {
            _logger.LogInformation($"Null sink {Id} stopped.");
        }
    }
}
```

In this example, the `NullSink` sink class inherits from the `EventSink` class because it provides the ability to transform records into different serialization formats such as JSON and XML.

2. Compile the previously defined project.
3. Copy the assembly to the installation directory for Kinesis Agent for Windows.
4. Create or update an `appsettings.json` configuration file that uses the new sink, and place it in the installation directory for Kinesis Agent for Windows. For example, to use the `RandomSource` and `NullSink` custom plugins, you could use the following `appsettings.json` configuration file:

```
{
  "Sources": [
  {
 "Id": "MyRandomSource",
 "SourceType": "RandomSource",
 "Rate": "00:00:10",
 "Max": 50
  }

  ],
  "Sinks": [
  {
    "Id": "MyNullSink",
 "SinkType": "NullSink",
 "Format": "json"
  }
  ],
  "Pipes": [
    {
   "Id": "MyRandomToNullPipe",
   "SourceRef": "MyRandomSource",
   "SinkRef": "MyNullSink"
 }
  ]
}
```

This configuration creates a source that sends an instance of `RandomData` with a `RandomValue` set to a random number between 0 and 50 every 10 seconds. It creates a sink that transforms the incoming `RandomData` instances to JSON, logs that JSON, and then discards the instances. Be sure to include both example factories, the `RandomSource` source class, and the `NullSink` sink class in the previously defined project to use the example configuration file.

5. Stop and then start Kinesis Agent for Windows.
6. Check the current Kinesis Agent for Windows log file (usually located in the `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` directory) to ensure that there are no issues with the custom sink plugin.
7. Ensure that data is arriving at the desired AWS service. Because the example `NullSink` does not stream to an AWS service, you can verify the correct operation of the sink by looking for log messages indicating that records have been received.

For example, you can see a log file similar to the following:

```
2018-10-18 12:36:36.3647 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AWS.AWSEventSinkFactory.
2018-10-18 12:36:36.4018 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.PerformanceCounterSinkFactory.
2018-10-18 12:36:36.4018 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory MyCompany.MySinks.NullSinkFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.DirectorySourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.ExchangeSource.ExchangeSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Uls.UlsSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.WindowsSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory MyCompany.MySources.RandomSourceFactory.
2018-10-18 12:36:36.9601 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.Pipes.PipeFactory.
2018-10-18 12:36:37.4694 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AutoUpdate.AutoUpdateFactory.
2018-10-18 12:36:37.4807 Amazon.KinesisTap.Hosting.LogManager INFO Performance
 counter sink  started.
2018-10-18 12:36:37.6250 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink starting.
2018-10-18 12:36:37.6250 Amazon.KinesisTap.Hosting.LogManager INFO Connected source
 MyRandomSource to sink MyNullSink
2018-10-18 12:36:37.6333 Amazon.KinesisTap.Hosting.LogManager INFO Random source id
 MyRandomSource started with rate 10000.
2018-10-18 12:36:47.8084 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":14}.
2018-10-18 12:36:57.6339 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":5}.
2018-10-18 12:37:07.6490 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":9}.
2018-10-18 12:37:17.6494 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":47}.
2018-10-18 12:37:27.6520 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":25}.
2018-10-18 12:37:37.6676 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":21}.
```

```
2018-10-18 12:37:47.6688 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":29}.
2018-10-18 12:37:57.6700 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":22}.
2018-10-18 12:38:07.6838 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":32}.
2018-10-18 12:38:17.6848 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":12}.
2018-10-18 12:38:27.6866 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":46}.
2018-10-18 12:38:37.6880 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":48}.
2018-10-18 12:38:47.6893 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":39}.
2018-10-18 12:38:57.6906 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":18}.
2018-10-18 12:39:07.6995 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":6}.
2018-10-18 12:39:17.7004 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":0}.
2018-10-18 12:39:27.7021 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":3}.
2018-10-18 12:39:37.7023 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":19}.
```

If you are creating a sink that accesses AWS services, there are base classes that you might find helpful. For a sink that uses the `AWSBufferedEventSink` base class, see `Amazon.KinesisTap.AWS\CloudWatchLogsSink.cs` in the source code for Kinesis Agent for Windows.

Document History for Amazon Kinesis Agent for Microsoft Windows User Guide

API version: 2018-10-15

The following table describes changes to the *Amazon Kinesis Agent for Microsoft Windows User Guide* (this document).

Change	Description	Date
Major documentation update	Added instructions for MSI installation. Updated DirectorySource configuration and added WindowsEventLogPollingSource. For sink configuration, added Local Filesystem sync configuration; ProfileRefreshingAWSCredentialProvider; information about text decorations, resolving variables in sink attributes, configuring STS regional endpoints for sinks, configuring VPC endpoints, and configuring alternate proxy servers. For pipes, added configuration attributes.	February 23, 2021
Update to documentation	Updated topic to indicate that Amazon S3 location specifications are case-sensitive.	November 7, 2018
Initial release, version 1.0.0.115	First release of the Kinesis Agent for Windows User Guide.	November 5, 2018

AWS Glossary

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