AWS CloudTrail

User Guide

Version 1.0

AWS CloudTrail: User Guide

Copyright © 2014 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, Cloudfront, CloudTrail, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS CloudTrail?

Welcome to the *AWS CloudTrail User Guide*. With AWS CloudTrail you can get a history of AWS API calls and related events for your account. This includes calls made by using the AWS Management Console, AWS SDKs, command line tools, and higher-level AWS services.

You can identify which users and accounts called AWS for services that support CloudTrail, the source IP address the calls were made from, and when the calls occurred. You can integrate CloudTrail into applications using the API, automate trail creation for your organization, check the status of your trails, and control how administrators turn CloudTrail logging on and off.

Topics

How AWS CloudTrail Works

AWS CloudTrail captures AWS API calls and related events made by or on behalf of an AWS account and delivers log files to an Amazon S3 bucket that you specify. Using CloudTrail's console in the AWS Management Console, the AWS CLI, or the CloudTrail API, you create a *trail*, which specifies the bucket for log file delivery and storage. By default, your log files are encrypted using Amazon S3 server-side encryption (SSE). You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. You can also optionally configure AWS CloudTrail to deliver events to a log group to be monitored by CloudWatch Logs.

CloudTrail typically delivers log files within 15 minutes of an API call. In addition, the service publishes new log files multiple times an hour, usually about every five minutes. These log files contain API calls from all of the account's services that support CloudTrail. For a list of AWS services that support CloudTrail, see Supported Services (p. 4).

> **Note**
> AWS CloudTrail records API calls made on an AWS account directly by the user or on behalf of the user by an AWS service. Examples of services that make API calls on behalf of users include, but are not limited to, AWS CloudFormation, AWS Elastic Beanstalk, AWS OpsWorks, and Auto

Scaling. For example, a AWS CloudFormation `CreateStack` call can result in additional API calls to Amazon EC2, Amazon RDS, Amazon EBS or other services as prescribed in the AWS CloudFormation template. This behavior is normal and expected. CloudTrail logs all of these API calls and provides a history of the calls made by the users directly or made by an AWS service as a result of the calls made by the user. You can identify the latter type of API call by examining the **invokedBy** field in the CloudTrail event.

You can choose to have CloudTrail publish SNS notifications when new log files are delivered, if you'll want to take quick action upon log file delivery. For information, see Configuring Amazon SNS Notifications (p. 29). You can also choose to have CloudTrail receive SNS notifications that you configured for CloudWatch alarms.

You can aggregate log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket (p. 32). Similarly, you can aggregate CloudTrail events across all regions to a CloudWatch Logs log group in one region. However, you cannot aggregate CloudTrail events from different AWS accounts into one CloudWatch Logs log group belonging to one AWS account.

There is no additional charge for CloudTrail, but standard rates for Amazon S3 and Amazon SNS apply. For more information about Amazon S3 rates, see Amazon S3 Pricing. For more information about Amazon SNS rates, see Amazon SNS Pricing.

CloudTrail Concepts

This section summarizes basic concepts related to CloudTrail.

Creating a Trail

Creating a trail means setting the configuration options to start logging AWS API calls and related events. That is, you must turn on the CloudTrail service, set up the target Amazon S3 bucket, (optionally) set up a log group for CloudWatch Logs to monitor log events, and (optionally) create an Amazon SNS topic to deliver CloudTrail notifications to you.

CloudTrail Console

The AWS CloudTrail console is a web application that you can use to manage the CloudTrail service. The console provides a user interface for performing many CloudTrail tasks such as turning on or editing CloudTrail, selecting an Amazon S3 bucket, setting a prefix, including or preventing API calls from global services such as IAM and AWS STS, and receiving Amazon SNS notifications for log file deliveries. For more information about the AWS management console in general, see AWS Management Console .

CloudTrail CLI

The AWS Command Line Interface is a unified tool that enables you to act easily with CloudTrail from the command line. For more information, see the CLI User Guide . For a complete list of the available CloudTrail CLI commands, see Available Commands .

CloudTrail APIs

In addition to the console and the CLI, you can also use the CloudTrail RESTful APIs to program CloudTrail directly. For more information see the AWS CloudTrail API Reference .

AWS SDKs

As an alternative to using the CloudTrail API, you can use one of the AWS SDKs. Each SDK consists of libraries and sample code for various programming languages and platforms. The SDKs provide a convenient way to create programmatic access to CloudTrail. For example, the SDKs take care of cryptographically signing requests, managing errors, and retrying requests automatically. For more information, see the Tools For AWS page.

IAM and CloudTrail

AWS Identity and Access Management is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions. Without IAM, organizations with multiple users and systems must either create multiple AWS accounts, each with its own billing and subscriptions to AWS products, or employees must all share the security credentials of a single AWS account. Also, without IAM, you have no control over the tasks a particular user or system can do and what AWS resources they might use.

Use IAM to create individual users for anyone who needs access to AWS CloudTrail. Create an IAM user for yourself as well, give that IAM user administrative privileges, and use that IAM user for all your work. By creating individual IAM users for people accessing your account, you can give each IAM user a unique set of security credentials. You can also grant different permissions to each IAM user. If necessary, you can change or revoke an IAM user's permissions any time. For more information, see Controlling User Access to AWS CloudTrail Actions (p. 24).

CloudWatch Logs and CloudTrail

Amazon CloudWatch is a web service that collects and tracks metrics to monitor in real time your Amazon Web Services (AWS) resources and the applications that you run on Amazon Web Services (AWS). Amazon CloudWatch Logs is a feature of CloudWatch that you can use specifically to monitor log data. Integration with CloudWatch Logs enables CloudTrail to send events containing API activity in your AWS account to a CloudWatch Logs log group. CloudTrail events that are sent to CloudWatch Logs can trigger alarms according to the metric filters you define. You can optionally configure CloudWatch alarms to send notifications or make changes to the resources that you are monitoring based on log stream events that your metric filters extract. Using CloudWatch Logs, you can also track CloudTrail events alongside events from the operating system, applications, or other AWS services that are sent to CloudWatch Logs.

Regional and Global Services

CloudTrail is a regional service. It creates trails in each region separately. By default, these trails include information for events that occur in those regions, plus events from global services such as IAM and AWS STS. For example, if you have two trails, each in a different region, and if you create a new IAM user, the create-user event is added to the log information in both regions.

If you configure CloudTrail to aggregate trail information from multiple regions in your account into a single Amazon S3 bucket, IAM events will be duplicated in the logs. The trail for each region will write the same IAM event to the aggregated log. To prevent this duplication, you can include global events selectively. A typical approach is to enable global events in one trail and to disable global events in other trails that write to the same Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket (p. 32).

How Does CloudTrail Relate to Other AWS Monitoring Services?

CloudTrail adds another dimension to the monitoring capabilities already offered by AWS; it does not change or replace logging features you might already be using. For example, where Amazon CloudWatch focuses on performance monitoring and system health, CloudTrail focuses on API activity. CloudTrail does not report on system performance or health, nor does it alter how you receive logs from your Amazon S3 or Amazon CloudFront subscriptions.

Partner Solutions

AWS partners with third-party specialists in logging and analysis to provide solutions that leverage CloudTrail output. For more information, visit the CloudTrail detail page at AWS CloudTrail.

CloudTrail Workflow

Here are the steps you take to use CloudTrail, which are described in detail in this guide:

1. Using the console, the AWS CLI, or the CloudTrail API, you create a trail, which consists of the information that CloudTrail uses to deliver log files to your Amazon S3 bucket or CloudWatch Logs log group. For information, see Creating and Updating Your Trail (p. 16).
2. (Optional) You create an Amazon SNS topic to which you subscribe for notifications that a new log file has arrived in your bucket. Amazon SNS can notify you in multiple ways, including programmatically using Amazon Simple Queue Service. For information, see Configuring Amazon SNS Notifications (p. 29).
3. You use the Amazon S3 API or console to retrieve the log files. For information, see Getting Your CloudTrail Log Files (p. 27).
4. You use the CloudTrail API, AWS CLI, or console to update your trail.
5. (Optional) You can use AWS Identity and Access Management to control which AWS users can create, configure, or delete trails, start and stop logging, and access the buckets that contain log information. For information, see Controlling User Access to AWS CloudTrail Actions (p. 24).
6. (Optional) You can use CloudWatch Logs to monitor events from API activity captured by CloudTrail. If so, you also create an IAM role with permissions for CloudTrail to send events to a CloudWatch Logs log group for monitoring. For more information, see [Monitoring CloudTrail Log Files by Using Amazon CloudWatch Logs.
7. (Optional) You can analyze your CloudTrail output by using one of the partner solutions that integrate with CloudTrail. These solutions offer a broad set of capabilities, such as change tracking, troubleshooting, and security analysis. For more information, visit the Amazon CloudTrail page.

Supported Services

This topic lists the services that currently support CloudTrail.

Supported Services

Analytics

- **Amazon Elastic MapReduce** (Supported beginning 04/04/2014)

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to process large amounts of data efficiently. Amazon EMR uses Hadoop processing combined with several services from AWS to perform such tasks as web indexing, data mining, log file analysis, machine learning, scientific simulation, and data warehousing. For more information, see the Amazon EMR Developer Guide. For more information about the Amazon EMR calls logged by CloudTrail, see Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail.

- **Amazon Kinesis** (Supported beginning 04/25/2014)

  Amazon Kinesis is a managed service that scales elastically for real-time processing of streaming big data. The service takes in large streams of data records that can then be consumed in real time by multiple data-processing applications that can be run on Amazon EC2 instances. For more information, see the Amazon Kinesis Developer Guide. For more information about the Amazon Kinesis calls logged by CloudTrail see Logging Amazon Kinesis API Calls by using AWS CloudTrail.

App Services

- **Amazon Simple Workflow Service** (Supported beginning 05/13/2014)

  The Amazon Simple Workflow Service (Amazon SWF) makes it easy to build applications that coordinate work across distributed components. In Amazon SWF, a task represents a logical unit of work that is performed by a component of your application. Coordinating tasks across the application involves managing intertask dependencies, scheduling, and concurrency in accordance with the logical flow of the application. Amazon SWF gives you full control over implementing tasks and coordinating them without worrying about underlying complexities such as tracking their progress and maintaining their state. For more information, see the Amazon SWF Developer Guide. For more information about the Amazon SWF calls logged by CloudTrail, see Logging Amazon Simple Workflow Service API Calls with AWS CloudTrail.
- **Amazon Simple Queue Service** (Supported beginning 07/16/2014)

  Amazon Simple Queue Service (Amazon SQS) offers reliable and scalable hosted queues for storing messages as they travel between computers. By using Amazon SQS, you can move data between distributed components of your applications that perform different tasks without losing messages or requiring each component to be always available. For more information, see the Amazon Simple Queue Service Developer Guide. For more information about the Amazon SQS calls logged by CloudTrail, see Logging Amazon SQS API Calls By Using AWS CloudTrail.
- **Amazon CloudSearch** (Supported beginning 10/16/2014)

  Amazon CloudSearch is a fully-managed service in the cloud that makes it easy to set up, manage, and scale a search solution for your website. Amazon CloudSearch enables you to search large collections of data such as web pages, document files, forum posts, or product information. For more information about Amazon CloudSearch, see the Amazon CloudSearch Developer Guide. For more information about the CloudSearch calls logged by CloudTrail, see Logging Amazon CloudSearch Configuration Service Calls Using AWS CloudTrail.
- **Amazon Elastic Transcoder** (Supported beginning 10/27/2014)

  Amazon Elastic Transcoder lets you convert media files that you have stored in Amazon S3 into media files in the formats required by consumer playback devices. For more information about Elastic Transcoder, see Amazon Elastic Transcoder Developer Guide. For more information about the Elastic Transcoder calls logged by CloudTrail, see Logging Elastic Transcoder API Calls Using CloudTrail.

Applications

- **Amazon Zocalo** (Supported beginning 08/26/2014)

Amazon Zocalo is a fully managed enterprise storage and sharing service. Your files are stored in the cloud safely and securely. Amazon Zocalo also includes a synchronization application that keeps selected folders on your local computer in sync with your files in the cloud. Your files are visible to only you and to your designated contributors and viewers. For more information about Amazon Zocalo, see Amazon Zocalo Administration Guide. For more information about the Amazon Zocalo actions logged by CloudTrail, see Logging Amazon Zocalo API Calls By Using CloudTrail.

Compute and Networking

- **AWS Direct Connect** (Supported beginning 03/08/2014)

  You can use AWS Direct Connect to establish a direct connection from your premises to AWS. This may reduce your network costs and increase bandwidth throughput. For more information about AWS Direct Connect, see the AWS Direct Connect User Guide. For more information about the AWS Direct Connect calls logged by CloudTrail, see Logging AWS Direct Connect API Calls in AWS CloudTrail.
- **Amazon Elastic Compute Cloud (EC2)** (Supported beginning 11/13/2013)

  Amazon Elastic Compute Cloud (Amazon EC2) provides resizeable computing capacity in the AWS cloud. You can launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 can also scale up or down quickly to handle changes in requirements or spikes in popularity, thereby reducing your need to forecast server traffic. For more information about Amazon EC2, see the Amazon EC2 User Guide. For a complete list of the Amazon EC2 calls logged by CloudTrail, see the Amazon EC2 API Reference.
- **Elastic Load Balancing** (Supported beginning 04/04/2014)

  You can use Elastic Load Balancing to automatically distribute your incoming application traffic across multiple Amazon EC2 instances. Elastic Load Balancing automatically scales request handling capacity in response to incoming traffic. For more information about Elastic Load Balancing, see the Elastic Load Balancing Developer Guide. For more information about the Elastic Load Balancing calls logged by CloudTrail, see Logging Elastic Load Balancing API Calls Using AWS CloudTrail.
- **Amazon Virtual Private Cloud** (Supported beginning 11/13/2013)

  Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you would operate in your own data center with the added benefit of using the scalable AWS infrastructure. For more information, see the Amazon Virtual Private Cloud User Guide. For a complete list of the Amazon VPC calls logged by CloudTrail, see Amazon Elastic Compute Cloud API Reference.
- **Auto Scaling** (Supported beginning 07/16/2014)

  Auto Scaling is a web service that enables you to automatically launch or terminate Amazon Elastic Compute Cloud (Amazon EC2) instances based on user-defined policies, health status checks, and schedules. For more information, see the Auto Scaling Developer Guide. For a list of the Auto Scaling calls logged by CloudTrail, see Logging Auto Scaling API Calls By Using CloudTrail.

Database

- **Amazon Relational Database Service** (Supported beginning 11/13/2013)

  Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizeable capacity for an industry-standard relational database and manages common database administration tasks. For more information, see the Amazon RDS User Guide. For more information about the Amazon RDS calls logged by CloudTrail, see Logging Amazon RDS API Calls Using CloudTrail.
- **Amazon Redshift** (Supported beginning 11/13/2013)

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data by using your existing business intelligence tools. It is optimized for datasets that range from a few hundred gigabytes to a petabyte or more. An Amazon Redshift data warehouse is a collection of computing resources called nodes which are organized into groups called clusters. Each cluster runs an Amazon Redshift engine and contains one or more databases. For more information, see the Amazon Redshift Developer Guide. For a complete list of the Amazon Redshift actions logged by CloudTrail, see the Amazon Redshift API Documentation.

- **Amazon ElastiCache** (Supported beginning 09/15/2014)

  Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale distributed in-memory cache environments in the cloud. It provides a high performance, resizeable, and cost-effective in-memory cache, while removing the complexity associated with deploying and managing a distributed cache environment. For more information, see the Amazon ElastiCache User Guide. For more information about the ElastiCache calls logged by CloudTrail, see Logging Amazon ElastiCache API Calls Using AWS CloudTrail.

Deployment and Management

- **AWS CloudFormation** (Supported beginning 04/02/2014)

  AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. It helps you leverage AWS products such as Amazon EC2, Amazon EBS, Amazon SNS, Elastic Load Balancing, and Auto Scaling to build highly reliable, highly scalable, cost-effective applications without worrying about creating and configuring the underlying AWS infrastructure. For more information, see the AWS CloudFormation User Guide. For more information about the AWS CloudFormation calls logged by CloudTrail, see Logging AWS CloudFormation API Calls in AWS CloudTrail.
- **AWS CloudTrail** (Supported beginning 11/13/2013)

  Like any supported service, when logging is turned on, CloudTrail logs actions to an Amazon S3 bucket that you specify. For a complete list of the actions that can be logged, see the CloudTrail API Reference.
- **AWS Elastic Beanstalk** (Supported beginning 03/31/2014)

  You can use AWS Elastic Beanstalk to quickly deploy and manage applications in the AWS cloud without worrying about the infrastructure that runs those applications. For more information, see the AWS Elastic Beanstalk Developer's Guide. For more information about the AWS Elastic Beanstalk calls logged by using CloudTrail, see Using AWS Elastic Beanstalk with AWS CloudTrail.
- **AWS Identity and Access Management** (Supported beginning 11/13/2013)

  AWS Identity and Access Management (IAM) is a web service that enables AWS customers to manage users and user permissions. By using IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access. For more information, see the IAM User Guide. For more information about the IAM calls logged by CloudTrail, see Logging AWS API Calls with AWS CloudTrail.
- **AWS Security Token Service** (Supported beginning 11/13/2013)

  You can use the AWS Security Token Service (STS) to grant a trusted user temporary, limited access to your AWS resources. For more information, see the AWS Security Token Service User Guide. For a complete list of AWS STS calls logged by CloudTrail, see the AWS Security Token Service API Reference.
- **Amazon CloudWatch** (Supported beginning 04/30/2014)

  Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics which are the variables you want to measure for your resources and applications. CloudWatch alarms send notifications or automatically make changes

to the resources you are monitoring based on rules that you define. For more information about CloudWatch, see the Amazon CloudWatch Developer Guide. For more information about the list of CloudWatch calls logged by CloudTrail, see Logging Amazon CloudWatch API Calls in AWS CloudTrail.

- **AWS OpsWorks** (Supported beginning 06/04/2014)

  AWS OpsWorks provides a simple and flexible way to create and manage stacks and applications. It supports a standard set of components—including application servers, database servers, load balancers, and more—that you can use to assemble your stack. These components all come with a standard configuration and are ready to run. For more information, see the AWS OpsWorks User Guide. For more information about the list of AWS OpsWorks calls logged by CloudTrail, see Logging AWS OpsWorks API Calls By Using AWS CloudTrail.

- **AWS Key Management Service** (Supported beginning 11/12/2014)

  The AWS Key Management Service is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data. AWS KMS is integrated with other AWS services including Amazon EBS, Amazon S3, and Amazon Redshift. For more information, see the AWS Key Management Service Developer Guide. For more information about the list of AWS KMS calls logged by CloudTrail, see Logging AWS KMS API Calls.

Mobile Services

- **Amazon Simple Notification Service** (Supported beginning 10/09/2014)

  Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. For more information about Amazon SNS, see the Amazon Simple Notification Service Developer Guide. For more information about the Amazon SNS calls logged by CloudTrail, see Logging Amazon Simple Notification Service API Calls By Using AWS CloudTrail.

Storage and Content Delivery

- **Amazon Elastic Block Store** (Supported beginning 11/13/2013)

  Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes persist independently from the life of the instance. For more information, see the Amazon EBS User Guide. For more information about the Amazon EBS calls logged by CloudTrail, see the Amazon EC2 API Reference.

- **Amazon CloudFront** (Supported beginning 05/28/2014)

  Amazon CloudFront speeds up distribution of your static and dynamic web content to end users. CloudFront delivers your content through a worldwide network of edge locations. When an end user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency, so that content is delivered with the best possible performance. For more information, see the Amazon CloudFront Developer Guide. For more information about the CloudFront calls logged by CloudTrail, see Using AWS CloudTrail to Capture Requests Sent to the CloudFront API.

Supported Regions

AWS CloudTrail currently supports the following regions:

Region Name	Region	Endpoint	Protocol	AWS Account ID	Support Date
US East (Northern Virginia) Region	us-east-1	[illegible]	HTTPS	086441151436	11/13/2013
US West (Northern California) Region	us-west-1	[illegible]	HTTPS	388731089494	05/13/2014
US West (Oregon) Region	us-west-2	[illegible]	HTTPS	113285607260	11/13/2013
EU (Ireland) Region	eu-west-1	[illegible]	HTTPS	859597730677	05/13/2014
Asia Pacific (Sydney) Region	ap-southeast-2	[illegible]	HTTPS	284668455005	05/13/2014
Asia Pacific (Singapore) Region	ap-southeast-1	[illegible]	HTTPS	903692715234	06/30/2014
Asia Pacific (Tokyo) Region	ap-northeast-1	[illegible]	HTTPS	216624486486	06/30/2014
South America (Sao Paulo) Region	sa-east-1	[illegible]	HTTPS	814480443879	06/30/2014
EU (Frankfurt) Region	eu-central-1	[illegible]	HTTPS	035351147821	10/23/2014

When you create a trail, you specify an Amazon S3 bucket and the region from which CloudTrail will deliver log files. Initially, CloudTrail will include only log files from that region in your bucket. You can, however, configure CloudTrail to also include log files from additional regions. For more information, see Aggregating Log Files from Multiple Regions (p. 33).

Log File Examples

The following sections show selected example log entries for a few of the services that are supported by CloudTrail. A log file is made up of one or more records. The following topics do not typically display all of the records that a log file might contain. They typically show only the records for an action that started the creation of a log file.

Topics

EC2 Log Examples

Amazon Elastic Compute Cloud (Amazon EC2) provides resizeable computing capacity in the Amazon Web Services (AWS) cloud. You can launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 can also scale up or down quickly to handle changes in requirements or spikes in popularity, thereby reducing your need to forecast server traffic. For more information about Amazon EC2, see the Amazon EC2 User Guide.

The following log file record shows that an IAM user named Alice called the Amazon EC2 **StartInstances** action by using the **ec2-start-instances** CLI command for instance `i-ebeaf9e2`.

```
{
    "Records": [{
        "eventVersion": "1.0",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "accountId": "123456789012",
            "userName": "Alice"
        },
        "eventTime": "2014-03-06T21:22:54Z",
        "eventSource": "ec2.amazonaws.com",
        "eventName": "StartInstances",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "205.251.233.176",
        "userAgent": "ec2-api-tools 1.6.12.2",
        "requestParameters": {
            "instancesSet": {
                 "items": [{
                      "instanceId": "i-ebeaf9e2"
                 }]
            }
        },
        "responseElements": {
            "instancesSet": {
                "items": [{
                      "instanceId": "i-ebeaf9e2",
                      "currentState": {
                          "code": 0,
                          "name": "pending"
                      },
                      "previousState": {
                          "code": 80,
                          "name": "stopped"
                      }
                    }]
            }
        }
    },
    ... additional entries ...
```

```
 ]
}
```

The following log file record shows that an IAM user named Alice called the Amazon EC2 **StopInstances** action by using the **ec2-stop-instances** command in the CLI.

```
{
    "Records": [{
        "eventVersion": "1.0",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-03-06T21:01:59Z",
        "eventSource": "ec2.amazonaws.com",
        "eventName": "StopInstances",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "205.251.233.176",
        "userAgent": "ec2-api-tools 1.6.12.2",
        "requestParameters": {
            "instancesSet": {
                "items": [{
                    "instanceId": "i-ebeaf9e2"
                }]
            },
            "force": false
        },
        "responseElements": {
            "instancesSet": {
                "items": [{
                    "instanceId": "i-ebeaf9e2",
                    "currentState": {
                        "code": 64,
                        "name": "stopping"
                    },
                    "previousState": {
                        "code": 16,
                        "name": "running"
                    }
                }]
            }
        }
    },
    ... additional entries ...
 ]
}
```

The following log file record shows that the Amazon EC2 console back-end called the **CreateKeyPair** action in response to requests initiated by an IAM user. Note that the **responseElements** contain a hash of the key pair and that the key material has been removed by AWS.

```
{
    "Records": [{
    {
        "eventVersion": "1.0",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice",
            "sessionContext": {
                "attributes": {
                    "mfaAuthenticated": "false",
                    "creationDate": "2014-03-06T15:15:06Z"
                }
            }
        },
        "eventTime": "2014-03-06T17:10:34Z",
        "eventSource": "ec2.amazonaws.com",
        "eventName": "CreateKeyPair",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.64",
        "userAgent": "EC2ConsoleBackend, aws-sdk-java/Linux/x.xx.fleetxen
Java_HotSpot(TM)_64-Bit_Server_VM/xx",
        "requestParameters": {
            "keyName": "mykeypair"
        },
        "responseElements": {
            "keyName": "mykeypair",
            "keyFingerprint":
"30:1d:46:d0:5b:ad:7e:1b:b6:70:62:8b:ff:38:b5:e9:ab:5d:b8:21",
            "keyMaterial": "\u003csensitiveDataRemoved\u003e"
        },
    ... additional entries ...
    ]
}
```

IAM Log Examples

AWS Identity and Access Management is a web service that enables AWS customers to manage users and user permissions. The service is targeted at organizations with multiple users. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access. Without IAM, organizations with multiple users and systems must either create multiple AWS accounts, each with its own billing and subscriptions to AWS products, or employees must all share the security credentials of a single AWS account. Also, without IAM, you have no control over the tasks a particular user or system can do and what AWS resources they might use. For more information, see the IAM User Guide

The following log file record shows that an IAM user called the **CreateUser** action to create a new user named Bob.

```
{
 "Records": [{
```

```
  "eventVersion": "1.0",
  "userIdentity": {
   "type": "IAMUser",
   "principalId": "EX_PRINCIPAL_ID",
   "arn": "arn:aws:iam::123456789012:user/Alice",
   "accountId": "123456789012",
   "accessKeyId": "EXAMPLE_KEY_ID",
   "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:11:59Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "CreateUser",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.3.2 Python/2.7.5 Windows/7",
  "requestParameters": {
   "userName": "Bob"
  },
  "responseElements": {
   "user": {
    "createDate": "Mar 24, 2014 9:11:59 PM",
    "userName": "Bob",
    "arn": "arn:aws:iam::123456789012:user/Bob",
    "path": "/",
    "userId": "EXAMPLEUSERID"
   }
  }
 }]
}
```

The following log file record shows that an IAM user called the **AddUserToGroup** action to add Bob to the administrators group.

```
{
"Records": [
        {
        "eventVersion": "1.0",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice",
            "sessionContext": {
                "attributes": {
                    "mfaAuthenticated": "false",
                    "creationDate": "2014-03-25T18:45:11Z"
                }
            }
        },
        "eventTime": "2014-03-25T21:08:14Z",
        "eventSource": "iam.amazonaws.com",
        "eventName": "AddUserToGroup",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
```

```
        "userAgent": "AWSConsole",
        "requestParameters": {
            "userName": "Bob",
            "groupName": "admin"
        },
        "responseElements": null
    },
    ...additional entries
  ]
}
```

The following log file record shows that an IAM user called the **CreateRole** action to create a new IAM role. The API was called by using a CLI command.

```
{
    "Records": [{
        "eventVersion": "1.0",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/Alice",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2014-03-25T20:17:37Z",
        "eventSource": "iam.amazonaws.com",
        "eventName": "CreateRole",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-cli/1.3.2 Python/2.7.5 Windows/7",
        "requestParameters": {
            "assumeRolePolicyDocument": "{\n  \"Version\": \"2012-10-17\",\n
\"Statement\": [\n    {\n      \"Sid\": \"\",
            \n\"Effect\": \"Allow\",\n      \"Principal\": {\n        \"AWS\":
 \"arn:aws:iam::210987654321:root\"\n     },\n     \"Action\": \"sts:AssumeR
ole\"\n    }\n  ]\n}",
            "roleName": "TestRole"
        },
        "responseElements": {
            "role": {
                "assumeRolePolicyDocument": "%7B%0A%20%20%22Ver
sion%22%3A%20%222012-10-17%22%2C%0A%20%20%22State
ment%22%3A%20%5B%0A%20%20%20%20%7B%0A%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C%0A%20%20%20%20%20%20%22Ef
fect%22%3A%20%22Allow%22%2C%0A%20%20%20%20%20%20%22Princip
al%22%3A%20%7B%0A%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws%3Ai
am%3A%3A803981987763%3Aroot%22%0A%20%20%20%20%20%20%7D%2C%0A%20%20%20%20%20%20%22Ac
tion%22%3A%20%22sts%3AAssumeRole%22%0A%20%20%20%20%7D%0A%20%20%5D%0A%7D",
                "roleName": "TestRole",
                "roleId": "AROAIUU2EOWSWPGX2UJUO",
                "arn": "arn:aws:iam::123456789012:role/TestRole",
                "createDate": "Mar 25, 2014 8:17:37 PM",
                "path": "/"
            }
        }
```

```
    }]
}
```

Getting Started

CloudTrail enables you to retrieve a history of API calls and other events for your account. This includes calls and events made by the AWS Management Console and command line tools, by any of the AWS SDKs, or by other AWS services. The following topics discuss how to get started using CloudTrail.

Topics

Creating and Updating Your Trail

This topic walks you through creating a trail for your AWS account. You can create trails using the AWS CloudTrail console or using the AWS command line interface (CLI). Both methods follow the same steps:

1. Turn on CloudTrail.
2. Create a new Amazon S3 bucket for storing your log files, or specify an existing bucket where you want the log files delivered.
3. (Optional) Create a new Amazon SNS topic in order to receive notifications when new log files are delivered.

In addition, the following sections explain how to update a trail and how to stop and start CloudTrail delivery of log files.

Topics

Amazon S3 Bucket Naming Requirements

The Amazon S3 bucket that you use to store CloudTrail log files must have a name that conforms with naming requirements for non-US Standard regions. Amazon S3 defines a bucket name as a series of one or more labels, separated by periods, that adhere to the following rules:

- The bucket name can be between 3 and 63 characters long, and can contain only lower-case characters, numbers, periods, and dashes.
- Each label in the bucket name must start with a lowercase letter or number.
- The bucket name cannot contain underscores, end with a dash, have consecutive periods, or use dashes adjacent to periods.
- The bucket name cannot be formatted as an IP address (198.51.100.24).

Amazon S3 Bucket Policy

By default, all Amazon S3 buckets and objects are private. Only the resource owner, the AWS account that created the bucket, can access the bucket and any objects it contains. The resource owner can, however, choose to grant access permissions to other resources and users by writing an access policy. The following topics discuss the bucket policy necessary to enable CloudTrail to write log files to a bucket from supported regions:

Topics

Creating an Amazon S3 Bucket Policy by using the CloudTrail Console or CLI

When you specify an Amazon S3 bucket as the location for log file delivery, you must make sure that you attach a policy to the bucket that allows CloudTrail to work with the bucket. The easiest way to do this is to create the bucket using the CloudTrail console or the AWS CLI. When you do so, CloudTrail automatically attaches the policy to the bucket for you and fills in the following fields:

- The allowed SIDs.
- The name of the folder where the log files will be stored.
- The current and upcoming regions in which CloudTrail can operate.
- The bucket name.
- The optional prefix if you specified one at creation.
- The ID of the owning account.

Manually Editing the Bucket Policy

If you use an existing bucket, it is best to use a bucket that was created specifically for CloudTrail. Perform the following steps to attach the required policy to the bucket:

To attach the policy required by CloudTrail to an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console.
2. Select the bucket where you want CloudTrail to deliver your log files, and then click **Properties**.

3. Click **Permissions**.
4. Click **Edit Bucket Policy**.
5. Copy the following policy into the **Bucket Policy Editor** window and then substitute the correct names of your bucket, prefix, and account number for the placeholders indicated in italics. If you specified a prefix when you created your trail, be sure to include it here. The prefix is an optional addition to the Amazon S3 object key that helps create a folder-like organization in your bucket.

   The following policy enables CloudTrail to write log files to your bucket from any of the currently Supported Regions (p. 9).

   Caution
   If the existing bucket already has one or more policies attached to it, add the statements for CloudTrail access to that policy or policies. We recommend that you evaluate the resulting set of permissions to be sure that they are appropriate for the users who will be accessing the bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSCloudTrailAclCheck20131101",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::903692715234:root",
          "arn:aws:iam::859597730677:root",
          "arn:aws:iam::814480443879:root",
          "arn:aws:iam::216624486486:root",
          "arn:aws:iam::086441151436:root",
          "arn:aws:iam::388731089494:root",
          "arn:aws:iam::284668455005:root",
          "arn:aws:iam::113285607260:root",
          "arn:aws:iam::035351147821:root"
        ]
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::myBucketName"
    },
    {
      "Sid": "AWSCloudTrailWrite20131101",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::903692715234:root",
          "arn:aws:iam::859597730677:root",
          "arn:aws:iam::814480443879:root",
          "arn:aws:iam::216624486486:root",
          "arn:aws:iam::086441151436:root",
          "arn:aws:iam::388731089494:root",
          "arn:aws:iam::284668455005:root",
          "arn:aws:iam::113285607260:root",
          "arn:aws:iam::035351147821:root"
        ]
      },
      "Action": "s3:PutObject",
     "Resource": "arn:aws:s3:::myBucketName/[optional] prefix/AWSLogs/myAccountID/*",
```

```
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}
```

Typically, when you create a trail, you specify the region that CloudTrail will deliver log files from. However, CloudTrail can also aggregate log files from other regions and accounts into your Amazon S3 bucket as long as those regions have write access to your bucket. For more information, see Aggregating Log Files from Multiple Regions (p. 33). For more information about the regions supported by CloudTrail, see Supported Regions (p. 9). For more information about using the Amazon S3 console to create or edit bucket policies, see the topic Editing Bucket Permissions in the *Amazon Simple Storage Service Console User Guide.*

Troubleshooting Amazon S3 Bucket Policy Errors

If you decide to use an existing bucket when you turn on CloudTrail for a new region in a new or existing trail, you might receive an error that says there is a problem with the bucket policy. If so, it is likely that your bucket policy does not enable access for the new region. For example, you might receive this error if your bucket policy supports only the us-east-1 (Northern Virginia) and us-west-2 (Oregon) regions and you try to turn on your trail in ap-southeast-2 (Sydney). For more information about the regions currently supported by CloudTrail, see Supported Regions (p. 9).

To work around this problem, perform one of the following actions:

- Use the CloudTrail console or CLI to turn on CloudTrail and specify a new bucket. CloudTrail automatically attaches the correct policy to the bucket for you.
- For an existing bucket, follow the preceding steps in the Manually Editing the Bucket Policy (p. 17) section by copying the listed policy straight into your Amazon S3 policy editor, remembering to substitute the correct names of your bucket, prefix, and account number for the placeholders indicated in italics.
- Also for an existing bucket, you can manually enter in the Amazon S3 bucket policy only those regions that you want to support. For example, if your bucket already supports the us-east-1 (Northern Virginia) and us-west-2 (Oregon) regions and you want to add support for us-west-1 (Northern California) and ap-southeast-2 (Sydney), edit the **Principal** sections of the Amazon S3 bucket policy to include the following ARNs.

```
...
  "Principal": {
    "AWS": [
      "arn:aws:iam::086441151436:root",
      "arn:aws:iam::113285607260:root",
      "arn:aws:iam::388731089494:root",
      "arn:aws:iam::284668455005:root"
    ]
  },
...
```

Creating and Updating a Trail with the CloudTrail Console

The following steps create a CloudTrail trail, including an Amazon S3 bucket and an optional Amazon SNS topic to which you can subscribe for notifications that new log files are available. While you can use an existing bucket, we recommend that you create a new one. When you create a new bucket, CloudTrail creates the necessary IAM policies for you on the bucket and topic.

Any trail you create using the console has the name "Default." If you want to specify a name for your trail, use the `update-subscription` command from the CLI, which is described in the next section. Currently, you can create one trail for each region where the service is supported. If you want to replace a trail and create a new one, first use the `delete-trail` command from the CLI.

To create a CloudTrail trail using the AWS Management Console

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/home.
2. Click **Get Started**.
3. On the **Turn On CloudTrail** page, next to **Create a new S3 bucket?**, select **Yes** to create a new bucket or **No** to use an existing one.

   Note
   If you click **No**, the console options change to provide a drop-down list from which you can select an existing Amazon S3 bucket. If you use an existing bucket, you must manually edit the bucket policy to grant CloudTrail permission to write to it. You can only designate an existing bucket owned by the account under which the trail is created. See the section Amazon S3 Bucket Policy (p. 17) for information about manually editing the policy for a bucket.
4. In the **S3 bucket name** field, enter a name for the bucket you want to designate for log file storage. See the section Amazon S3 Bucket Naming Requirements (p. 17) for information about bucket naming rules and conventions.
5. (Optional) If you want to enter a prefix for your bucket, subscribe to global services such as IAM or AWS STS, or create an Amazon SNS topic, click **Advanced**.
6. (Optional) In the **Log file prefix** field, enter a prefix for your Amazon S3 bucket. The prefix is an addition to the URL for an Amazon S3 object that helps create a folder-like organization in your bucket. Hover your cursor over the sentence under the text field to see where your log files will be stored.
7. (Optional) Select **Yes** or **No** for **Include global services?** depending on whether or not you want to record API calls from global services such as IAM or AWS STS. In most circumstances, you should accept the default **Yes** setting. To learn about advanced scenarios in which it is appropriate to select **No**, see the topic Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket (p. 32).

   Note
   We recommend including global services. If you are aggregating trails from multiple regions into one bucket, we recommend that you include global services in only one trail. This avoids generating duplicate entries for the global events in the log files, which would otherwise occur because global services generate the same events in all regions.
8. (Optional) Select **Yes** or **No** for **SNS notification for every log file delivery?** If you select **Yes**, enter a name for your Amazon SNS topic in the **SNS topic (new)** field.

   Note
   If you create a topic, you must subscribe to it in order to get notification of log file delivery. Due to the frequency of notifications, we recommend that you configure the subscription to use an Amazon SQS queue to handle notifications programmatically. See the Amazon Simple Notification Service Getting Started Guide for information.

9. Click **Subscribe**.
10. In about 15 minutes, CloudTrail starts publishing log files that show the AWS API calls made in your account since you completed the preceding steps.

To update a trail using the AWS Management Console

1. Navigate to the **CloudTrail Configuration** page of the console.
2. Click **Edit** in the upper left corner.
3. Modify the settings that you want to update for your trail.

   Note
   When you update a trail, you can optionally configure CloudTrail to deliver events to CloudWatch Logs for real-time monitoring. For more information, see Sending CloudTrail Events to CloudWatch Logs (p. 50).

4. Click **Save**.

   Note
   We recommend that you use the same prefix that you used before when you designate a different bucket for your trail. Otherwise, you must manually update the bucket policy with the changed prefix.

Creating and Updating a Trail with the AWS CLI

You can create a trail using the `create-subscription` command. The command uses the following options to specify the additional settings for the trail:

- `--name` specifies the name of the trail.
- `--s3-use-bucket` specifies an existing Amazon S3 bucket for log file storage.
- `--s3-new-bucket` specifies the name of the new bucket created when the command executes.
- `--s3-prefix` specifies a prefix for the log file delivery path (optional).
- `--sns-new-topic` specifies the name of the Amazon SNS topic to which you can subscribe for notification of log file delivery to your bucket (optional).

As part of defining the CloudTrail trail, the `create-subscription` command can create a new Amazon S3 bucket for log file delivery and a new Amazon SNS topic for notifications. In contrast to trails created using the console, every trail created with the AWS CLI must have a name. You specify this name with command line operations. Again, you are limited to one trail per account for each region in which the account is running AWS resources.

Note
The AWS CLI commands shown below require that you have the AWS command line tools. For more information, see the AWS Command Line Interface User Guide.

The following example command demonstrates the creation of a trail for an account using the AWS CLI.

```
 aws cloudtrail create-subscription --name=awscloudtrail-example --s3-new-
bucket=awscloudtrail-new-bucket-example --s3-prefix=prefix-example --sns-new-
topic=awscloudtrail-example-log-deliverytopic
```

If the command executes successfully, you see output similar to the following:

```
CloudTrail configuration:
{
  "trailList": [
    {
      "S3KeyPrefix": "prefix-example",
      "IncludeGlobalServiceEvents": true,
      "Name": "awscloudtrail-example",
      "SnsTopicName": "awscloudtrail-example-log-deliverytopic",
      "S3BucketName": "awscloudtrail-new-bucket-example"
    }
  ]
}
```

You can update your trail using the command `update-subscription` and setting the options to new values. The following example designates a different Amazon S3 bucket. If you want a trail with a different name, you can delete the trail and run `create-subscription` again.

```
aws cloudtrail update-subscription --name=awscloudtrail-example --s3-use-buck
et=awscloudtrail-new-bucket-example2--s3-prefix=prefix-example
```

If the command executes successfully, you see that the `trailList` structure has been updated:

```
CloudTrail configuration:
{
  "trailList": [
    {
      "S3KeyPrefix": "prefix-example",
      "IncludeGlobalServiceEvents": true,
      "Name": "awscloudtrail-example",
      "SnsTopicName": "awscloudtrail-example-log-deliverytopic",
      "S3BucketName": "awscloudtrail-new-bucket-example2"
    }
  ]
}
```

Tip
If you designate an existing bucket for log file publication, we recommend that you use one that was created with the CloudTrail console or CLI, as it will already have the necessary policy applied. If you decide to use an existing bucket, however, see the topic Amazon S3 Bucket Policy (p. 17) for a procedure that shows you how to apply the necessary policy.

Additional AWS CLI Commands

The CloudTrail CLI includes several other commands that help you manage your trails. This section demonstrates how to use these commands.

Retrieving Trail Settings and the Status of a Trail

Use the following command to retrieve trail settings:

```
aws cloudtrail describe-trails
```

If the command succeeds, you see output similar to the following:

```
{
"trailList": [
    {
        "S3KeyPrefix": "prefix-example",
        "IncludeGlobalServiceEvents": true,
        "Name": "awscloudtrail-example",
        "SnsTopicName": "awscloudtrail-example-log-deliverytopic",
        "S3BucketName": "awscloudtrail-new-bucket-example2"
    }
]
}
```

Use the following command to retrieve the status of a trail:

```
aws cloudtrail get-trail-status --name awscloudtrail-example
```

If the command succeeds, you see output similar to the following:

```
{
"LatestDeliveryAttemptTime": "2013-11-12T20:18:27Z",
"LatestNotificationAttemptSucceeded": "2013-11-12T20:18:27Z",
"LatestDeliveryAttemptSucceeded": "2013-11-12T20:18:27Z",
"IsLogging": true,
"TimeLoggingStarted": "2013-11-12T20:19:30Z",
"LatestNotificationAttemptTime": "2013-11-12T20:18:27Z",
"TimeLoggingStopped": "2013-11-12T20:19:16Z"
}
```

In addition to the fields shown in the preceding JSON code, the status contains the following fields if there are Amazon SNS or Amazon S3 errors:

- `LatestNotificationError`. Contains the error emitted by Amazon SNS if a subscription to a topic fails.
- `LatestDeliveryError`. Contains the error emitted by Amazon S3 if CloudTrail cannot deliver a log file to a bucket.

Stopping and Starting Logging for a Trail

The following commands start and stop CloudTrail logging, respectively:

```
aws cloudtrail start-logging --name awscloudtrail-example
```

```
aws cloudtrail stop-logging --name awscloudtrail-example
```

Caution
Before deleting a bucket, you should use `stop-logging` to end all logging to the bucket. If you don't stop logging, CloudTrail will continue to attempt to deliver log files to a bucket of the same name for a limited period of time.

Deleting a Trail

You can delete a trail using the following command:

```
aws cloudtrail delete-trail --name awscloudtrail-example
```

When you delete a trail, you do not delete either the Amazon S3 bucket or the Amazon SNS topic associated with it. If you want to delete these items, do so separately using the AWS Management Console, AWS CLI, or service API.

Controlling User Access to AWS CloudTrail Actions

AWS CloudTrail integrates with AWS Identity and Access Management (IAM), which allows you to control access to CloudTrail and to other AWS resources that CloudTrail requires, including Amazon S3 buckets and Amazon Simple Notification Service (Amazon SNS) topics. You can use AWS Identity and Access Management to control which AWS users can create, configure, or delete AWS CloudTrail trails, start and stop logging, and access the buckets that contain log information.

If you work with CloudTrail as the root user in your account, you can perform all the tasks associated with trails, including creating trails, reading logs, and so on. If other people in your organization need to work with CloudTrail, you can create IAM users for those people and give them individual names and passwords. When you do that, you must also give users permissions to work with CloudTrail and with any other AWS services they need to access, such as Amazon S3. (By default, IAM users have no permissions and cannot perform any actions in AWS.)

> **Important**
> We consider it a best practice not to use root account credentials to perform everyday work in AWS. Instead, we recommend that you create an IAM administrators group with appropriate permissions, create IAM users for the people in your organization who need to perform administrative tasks (including for yourself), and add those users to the administrative group. For more information, see IAM Best Practices in the *Using IAM* guide.

Topics

Granting Permissions for CloudTrail Actions

To allow users to administer a CloudTrail trail, you must grant explicit permissions to IAM users to perform the actions associated with CloudTrail tasks. For most scenarios, you can do this using a policy template that contains predefined permissions.

> **Note**
> The permissions you grant to users to perform CloudTrail administration tasks are not the same as the permissions that CloudTrail itself requires in order to deliver log files to Amazon S3 buckets or send notifications to Amazon SNS topics. For more information about those permissions, see Getting Your CloudTrail Log Files (p. 27). CloudTrail also requires a role that it can assume to deliver events to an Amazon CloudWatch Logs log group. For more information, see Granting Custom Permissions (p. 25).

A typical approach is to create an IAM group that has the appropriate permissions and then add individual IAM users to that group. For example, you might create an IAM group for users who should have full access to CloudTrail actions, and a separate group for users who should be able to view trail information but not create or change trails.

To create an IAM group and users for CloudTrail access

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. From the dashboard, click **Create a New Group of Users**, enter a name for the group, and then click **Continue**.
3. On the **Permissions** page, choose **Select Policy Template** and then select one of the templates for CloudTrail:

   - **AWS CloudTrail Full Access**. This policy gives users in the group full access to CloudTrail actions and permissions to manage the Amazon S3 bucket, manage the log group for CloudWatch Logs, and Amazon SNS topic for a trail.
   - **AWS CloudTrail Read Only Access**. This policy lets users in the group view trails and view buckets.

   Note
   You can also create a custom policy that grants permissions to individual actions. For details, see Granting Custom Permissions (p. 25) .
4. On the **Users** page, add existing IAM users to the new group. If you don't already have IAM users, click the **Create New Users** tab and enter user names, and then click **Continue**.
5. If you created new users, click **Users** in the navigation pane and do the following for each user:

   a. Select the user.
   b. If the user will use the console to manage CloudTrail, in the **Security Credentials** tab, click **Manage Password**, and then create a password for the user.
   c. If the user will use the CLI or API to manage CloudTrail, and if you didn't already create access keys, in the **Security Credentials** tab, click **Manage Access Keys** and then create access keys. Store the keys in a secure location.
   d. Give each user his or her credentials (access keys or password).

Additional Resources

To learn more about creating IAM users, groups, policies, and permissions, see Creating an Admins Group Using the Console and Permissions and Policies in the *Using IAM* guide.

Granting Custom Permissions

For most scenarios, the CloudTrail policy templates grant appropriate permissions to users who work with CloudTrail. However, you might need to grant slightly different permissions to users. If so, you can start by using the policy template to attach a policy to an IAM group or to an individual user, and then edit the policy to include (or exclude) specific permissions. Or you can create a custom policy and write the policy yourself.

You grant permissions to users, or to groups that users are in, by attaching an IAM policy to the user or group. (If you used a policy template, you can select the group or user that the policy is attached to, and in the **Permissions** tab, click **Manage Policy**.) Policies are JSON documents that define what actions a user is allowed to perform and what resources the user is allowed to perform those actions on. The following example shows a policy that provides read-only access to CloudTrail trails. It grants permissions to see trail information, but not to create or update trails. The policy also grants permission to read objects in Amazon S3 buckets, but not create or delete them.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudtrail:DescribeTrails",
        "cloudtrail:GetTrailStatus",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    }
  ]
}
```

In the policy statements, the `Effect` element specifies whether the actions will be allowed (as here) or denied. The `Action` element lists the specific actions that the user is allowed to perform. The `Resource` element lists the AWS resources that the actions are permitted for. For policies that control access to CloudTrail actions, the `Resource` element is always set to `*`, a wildcard that means "all resources."

Because the policy does not grant permissions for the `CreateTrail`, `UpdateTrail`, `StartLogging` and `StopLogging` actions, the user would not be allowed to turn logging on and off.

The values in the `Action` element correspond to the APIs that the services support. To see what actions CloudTrail supports, see the AWS CloudTrail API Reference. The actions are preceded by `cloudtrail:` to indicate that they refer to CloudTrail actions. For convenience, you can use the `*` wildcard character in actions as well, as shown in the following examples:

- `"Action": ["cloudtrail:*Logging"]`

  This allows all CloudTrail actions that end with "Logging" (`StartLogging`, `StopLogging`).
- `"Action": ["cloudtrail:*"]`

  This allows all CloudTrail actions, but not actions for other AWS services.
- `"Action": ["*"]`

  This allows all AWS actions. This permission would be suitable for a user who acts as an AWS administrator for your account.

The following example shows a "full access" (administrator) policy for working with CloudTrail. This policy lets a user perform all CloudTrail actions. It also lets the user manage files in Amazon S3 buckets, manage CloudWatch Logs monitoring of CloudTrail log events, and manage Amazon SNS topics in the account that the user is associated with.

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

```
      "Action": [
        "sns:AddPermission",
        "sns:CreateTopic",
        "sns:DeleteTopic",
        "sns:ListTopics",
        "sns:SetTopicAttributes"
      ],
      "Resource": "arn:aws:sns:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetObject",
        "s3:ListAllMyBuckets",
        "s3:PutBucketPolicy"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "cloudtrail:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
      ],
      "Resource": "arn:aws:logs:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "iam:ListRoles",
        "iam:GetRolePolicy",
      ],
      "Resource": "arn:aws:iam::*"
    }
  ]
}
```

Additional Resources

To learn more about creating IAM users, groups, policies, and permissions, see Creating an Admins Group Using the Console and Permissions and Policies in the *Using IAM* guide.

Getting Your CloudTrail Log Files

After you've set up CloudTrail to capture the log files you want, you'll need to be able to find the log files and interpret the information they contain.

CloudTrail delivers your log files to an Amazon S3 bucket that you specify when you create the trail. Typically, log files appear in your bucket within 15 minutes of the recorded AWS API call or other AWS event. Log files are generally published every five minutes.

Topics

Finding Your Log Files

CloudTrail publishes log files to your S3 bucket in a gzip archive. Within the S3 bucket, the log file has a formatted name that includes the following elements:

- The bucket name that you specified when creating your trail
- The (optional) prefix you specified when creating your trail
- The string "AWSLogs"
- The account number
- The string "CloudTrail"
- A region identifier such as us-west-1
- The year the log file was published in `YYYY` format
- The month the log file was published in `MM` format
- The day the log file was published in `DD` format
- An alphanumeric string that disambiguates the file from others that cover the same time period.

The following example shows a complete log file object name:

```
bucket_name>/prefix_name/AWSLogs/Account ID/CloudTrail/re
gion/YYYY/MM/DD/file_name.json.gz
```

To retrieve a log file, you can use the Amazon S3 console, the Amazon S3 command line interface (CLI), or the API. For example, open the Amazon S3 console, click on the name of the bucket in which you're interested, and keep clicking through the object hierarchy until you get to the log file you're looking for. All log files have a .gz extension. You will be navigating through an object hierarchy that is similar to the following but with a different bucket name, account ID, region and date

```
All Buckets
    Bucket_Name
        AWSLogs
            123456789012
                CloudTrail
                    us-west-1
                        2014
                            06
                                20
```

A log file for the preceding object hierarchy will look like the following:

```
123456789012_CloudTrail_us-west-1_20140620T1255ZHdkvFTXOA3Vnhbc.json.gz
```

Note
Although it is quite rare, you may receive log files that contain one or more duplicate events. Duplicate events will have the same **eventID**. For more information about the **eventID** field, see Record Body Contents (p. 71).

Reading Your Log Files

CloudTrail log files are Amazon S3 objects. You can retrieve them by using the Amazon S3 console, the AWS command line interface (CLI), or the API. For more information, see Working with Amazon S3 Objects in the Amazon Simple Storage Service Developer Guide. The Amazon Simple Storage Service Console User Guide covers using the console to retrieve your objects. For example, open the Amazon S3 console, click on the name of the bucket in which you're interested, and keep clicking through the object hierarchy until you get to the log file you're looking for. All log files have a .gz extension.

Log files are written in JSON (JavaScript Object Notation) format. If you have a JSON viewer add-on installed, you can view the files directly in your browser by double-clicking the log file name in the Amazon S3 bucket. This will open a new window or a new tab, depending on the add-on and on the browser, that displays the JSON in a readable format. To find a JSON viewer, search on that phrase in your browser of choice.

For example, if you use Google Chrome in Windows or Linux, a popular extension that you can add is named JSONView which you can download from the Chrome Web Store. If you use Mozilla Firefox, you can also download the JSONView add-on. With JSONView, you can double-click the compressed .gz file in your Amazon S3 bucket to open the log file in JSON format. There is no comparable extension for Internet Explorer, but there is a registry edit you can make to enable Internet Explorer to open JSON files after you download and decompress them.

An alternate approach to viewing your CloudTrail logs on Windows is to download them locally and use a text editor such as Notepad++ along with the JSON Viewer plug-in. To download a log file, right-click on the file in your Amazon S3 bucket and right-click **Download** in the pop-up window. Click **Save link as...** and follow the prompts to save the file locally. This saves the file, however, in compressed format. You must use a product such as 7-Zip to extract the uncompressed JSON data. After decompressing the file, open it in Notepad++, select all of the text, and navigate to **Plugins**, point to **JSON Viewer**, and then click **Format JSON**.

For more information about the event fields that can appear in a log file entry, see CloudTrail Event Reference (p. 70).

AWS partners with third-party specialists in logging and analysis to provide solutions that leverage CloudTrail output. For more information, visit the CloudTrail detail page at http://aws.amazon.com/cloudtrail.

Configuring Amazon SNS Notifications

You can be notified when CloudTrail publishes new log files to your Amazon S3 bucket. You manage notifications using Amazon Simple Notification Service (Amazon SNS).

Notifications are optional. If you want notifications, you configure CloudTrail to send update information to an Amazon SNS topic whenever a new log file has been sent. To receive these notifications, you can use Amazon SNS to subscribe to the topic. As a subscriber you can get updates sent to a Amazon Simple Queue Service (Amazon SQS) queue, which enables you to handle these notifications programmatically.

Topics

Configuring CloudTrail to Send Notifications

When you set up a trail, CloudTrail can configure a new Amazon SNS topic for you. If you create your trail using either the CloudTrail console or the `aws cloudtrail create-subscription` CLI command, and if you specify that you want notifications, CloudTrail creates the Amazon SNS topic for you and attaches an appropriate policy to it to allow CloudTrail to publish to that topic.

A CloudTrail trail works in a specific region. When you enable notifications, notifications are sent to an Amazon SNS topic in that region. If you have CloudTrail enabled for multiple regions, you need to create a separate topic for each region and subscribe to them individually.

CloudTrail lets you configure a trail to send notifications to an Amazon SNS topic, but it does not manage subscriptions for you. In order to receive notifications, you must subscribe to the Amazon SNS topic or topics that CloudTrail uses. You do this using the Amazon SNS console or Amazon SNS commands. For information, see Subscribe to a Topic in the *Amazon Simple Notification Service Developer Guide*.

> **Tip**
> Because CloudTrail sends a notification each time a log file is written to the Amazon S3 bucket, an account that's very active can generate a large number of notifications. If you subscribe using email or SMS, you can end up receiving a large volume of messages. We recommend that you subscribe using Amazon Simple Queue Service (Amazon SQS), which lets you handle notifications programmatically. For more information, see Subscribe Queue to Amazon SNS Topic in the *Amazon Simple Queue Service Developer Guide*.

The Amazon SNS notification consists of a JSON object that includes a **Message** field. The **Message** field lists the full path to the log file, as shown in the following example:

```
"Message" :   "{\"s3Bucket\":\"<your_bucket_name>\",\"s3ObjectKey\":
[\"AWSLogs/279630728954/CloudTrail/us-west-2/2013/12/13/<your_account_num
ber>_CloudTrail_us-west-2_20131213T1920Z_LnPgDQnpkSKEsppV.json.gz\"]}",
```

If you choose to receive notifications by email, the body of the email consists of the content of the **Message** field. For a complete discussion of the JSON structure, see the topic Sending Amazon SNS Messages to Amazon SQS Queues in the Amazon Simple Notification Service Developer Guide. Only the **Message** field conveys CloudTrail information. The other fields contain information from the Amazon SNS service itself.

If you create a trail using the API, you can specify an existing Amazon SNS topic that you want CloudTrail to send notifications to. In that case, you must make sure that the topic exists and that it has permissions that let CloudTrail send notifications to it. (See below.) You can specify that you want CloudTrail to use an existing Amazon SNS topic by calling the `CreateTrail` or `UpdateTrail` operations, which are part of the CloudTrail API.

Additional Resources

To learn more about creating Amazon SNS topics and about subscribing to them, see the Amazon Simple Notification Service Developer Guide.

Permissions for SNS Notifications

CloudTrail must have permissions to send notifications to an Amazon SNS topic. If CloudTrail creates the topic for you automatically (for example, if you use the console to set up a new trail or use the `aws cloudtrail create-subscription` command), these permissions are automatically added to the new topic. However, if you specify an existing topic, you must make sure that the topic has the correct permissions.

The following example shows the permissions that are automatically created by CloudTrail for a new topic. This policy statement allows CloudTrail to publish to a specified Amazon SNS topic.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AWSCloudTrailSNSPolicy20140219",
    "Effect": "Allow",
    "Principal": {"AWS": [
      "arn:aws:iam::903692715234:root",
      "arn:aws:iam::859597730677:root",
      "arn:aws:iam::814480443879:root",
      "arn:aws:iam::216624486486:root",
      "arn:aws:iam::086441151436:root",
      "arn:aws:iam::388731089494:root",
      "arn:aws:iam::284668455005:root",
      "arn:aws:iam::113285607260:root",
      "arn:aws:iam::035351147821:root"
    ]},
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:region:account-number:topic-name"
  }]
}
```

In the `Resource` field, *`account-number`* is the account number of the topic owner; in topics that you create, this will be your account number. You must substitute appropriate values for *`region`* and *`topic-name`*.

In the `Principal` field, the account number must match the CloudTrail account for that region. For more information about the supported regions and their associated account numbers, see Supported Regions (p. ?).

Additional Resources

To learn more about creating Amazon SNS topics and about subscribing to them, see the Amazon Simple Notification Service Developer Guide.

Working with Log Files

The following topics discuss how to perform more complex actions with your CloudTrail files. You can aggregate your log files into a single S3 bucket from multiple regions or accounts. You can share customer log files between accounts. You can sign CloudTrail log files to ensure that they have not been changed after delivery. You can monitor CloudTrail log files in real time by sending them to CloudWatch Logs. You can also use the AWS CloudTrail Processing Library to write log processing applications in Java.

Topics

Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket

With AWS CloudTrail, you can choose to have log files from multiple AWS regions and multiple AWS accounts delivered to a single Amazon S3 bucket. Aggregating your log files in this way simplifies storing and managing them. For more information, see the following topics.

Topics

Turning On CloudTrail for the First Time

Use the CloudTrail console to turn on the service for the first time.

To turn on CloudTrail

1. Sign into the AWS management console and open the AWS CloudTrail console at https://console.aws.amazon.com/cloudtrail/. In the navigation bar, select the region where you want to turn on CloudTrail.

2. Click **Get Started**.
3. On the **Turn On CloudTrail** page, select **Yes** or **No** for **Create a new S3 bucket?**. Select **Yes** to have a new Amazon S3 bucket created in your current account. Select **No** to display more options so that you can select an existing Amazon S3 bucket in your current account, or search for and select an existing bucket that is not in your account. If you select **No**, remember that you must be sure to manually edit the bucket policy to grant CloudTrail permission to write to it.

   For more information about bucket policy when aggregating log files from multiple AWS regions, see Setting Bucket Policy (p. 34)
4. For **S3 bucket name**, accept the suggested default or enter a name for the bucket for your log files.

   Note
   You must adhere to Amazon S3 bucket naming conventions. For more information about bucket naming, see the section .
5. (Optional) If you want to enter a prefix for your bucket, subscribe to global services such as IAM or AWS STS, or create an Amazon SNS topic, click **Advanced**.
6. (Optional) For **Log file prefix**, accept the suggested default or enter a prefix for your Amazon S3 bucket. The prefix is an addition to the URL for an Amazon S3 object that helps create a folder-like organization in your bucket.

   Tip
   Hover your cursor over **View log file location** to see where your log files will be stored.
7. (Optional) Select **Yes** for **Include global services?** to record API calls from global services such as IAM or AWS STS.
8. (Optional) Select **Yes** or **No** for **SNS notification for every log file delivery?**. If you select **Yes**, enter a name for your Amazon SNS topic in the **SNS topic (new)** field.

   Note
   If you create a topic, you must subscribe to it in order to get notified of log file delivery. Because notifications are frequent, you might want to configure the subscription to use an Amazon SQS queue to handle notifications programmatically. For more information, see the Amazon Simple Notification Service Getting Started Guide.
9. Click **Subscribe**.
10. In about 15 minutes, CloudTrail starts publishing log files that show the AWS API calls made in your accounts since you completed the preceding steps.

Aggregating Log Files from Multiple Regions

To aggregate CloudTrail log files from multiple regions into a single Amazon S3 bucket, you must complete the following steps in order.

1. Turn on CloudTrail in your AWS account, making sure to do so in only one region. When you turn on CloudTrail in your AWS account, it's a good idea to accept the default and choose to create a new Amazon S3 bucket for your log files. Then, when you turn on CloudTrail in additional regions, you can use the same log file prefix that you used when you turned on CloudTrail in the original region. This way, your log files from all regions are aggregated into a single bucket without your having to manually update the bucket permissions. For more infirmation, see Turning On CloudTrail for the First Time (p. 32).
2. Ensure that the bucket policy on your destination bucket grants the necessary permissions to CloudTrail. If you chose to create a new Amazon S3 bucket during the preceding step, the necessary permissions are granted for you. If, however, you specified an existing bucket during the previous step, you must grant the necessary permissions by applying a bucket policy. For more information, see Setting Bucket Policy (p. 34).

3. Turn on CloudTrail in the other regions where you have AWS resources, and then configure CloudTrail to use the same bucket and the same log file prefix that you specified in step 1. For more information, see Turning On CloudTrail in Additional Regions (p. 35).

Topics

Setting Bucket Policy

This procedure is necessary **only** if you chose to use an existing Amazon S3 bucket when you first turned on the CloudTrail service. If you chose to create a new bucket when you turned on CloudTrail, then the CloudTrail service has already applied the permissions necessary to deliver log files to the destination bucket.

To attach the policy required by CloudTrail to an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the bucket where you want CloudTrail to deliver your log files and then click **Properties**.
3. Click **Permissions**.
4. Click **Edit Bucket Policy**.
5. Copy the following policy into the **Bucket Policy Editor** window and then substitute the correct names of your bucket, prefix, and account ID for the placeholders indicated in italics. Your AWS account ID is a twelve-digit number, and leading zeros must not be omitted.

   Caution
   If the existing bucket already has one or more policies attached to it, add the statements for CloudTrail access to that policy or policies. Take care to evaluate the resulting set of permissions to be sure that they are appropriate for the users who will be accessing the bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSCloudTrailAclCheck20131101",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::903692715234:root",
          "arn:aws:iam::859597730677:root",
          "arn:aws:iam::814480443879:root",
          "arn:aws:iam::216624486486:root",
          "arn:aws:iam::086441151436:root",
          "arn:aws:iam::388731089494:root",
          "arn:aws:iam::284668455005:root",
          "arn:aws:iam::113285607260:root",
          "arn:aws:iam::035351147821:root"
        ]
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::myBucketName"
```

```
    },
    {
      "Sid": "AWSCloudTrailWrite20131101",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::903692715234:root",
          "arn:aws:iam::859597730677:root",
          "arn:aws:iam::814480443879:root",
          "arn:aws:iam::216624486486:root",
          "arn:aws:iam::086441151436:root",
          "arn:aws:iam::388731089494:root",
          "arn:aws:iam::284668455005:root",
          "arn:aws:iam::113285607260:root",
          "arn:aws:iam::035351147821:root"
        ]
      },
      "Action": "s3:PutObject",
     "Resource": "arn:aws:s3:::myBucketName/[optional] myLogFilePrefix/AWS
Logs/myAccountID/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}
```

Note
In the `Principal` fields, the account number must match the CloudTrail account for that region. For more information about the supported regions and their associated account numbers, see Supported Regions (p. ?).

Turning On CloudTrail in Additional Regions

Turning on CloudTrail in additional regions is only slightly different from the steps you took to turn it on initially.

To turn on CloudTrail in additional regions

1. Sign into the AWS management console using an account other than that which you used to initially sign in. Open the AWS CloudTrail console. In the navigation bar, select the region where you want to turn on CloudTrail. This should be a different region than that which you chose when you initially created a trail.
2. Click **Get Started** or, if you have already turned on CloudTrail, click **Edit**.
3. On the following page, for **Create a new S3 bucket?**, select **No**. Select an S3 bucket that exists in the current account or use the text box to enter the name of an existing bucket that is not in your current account. Remember that you must manually edit the bucket policy to grant CloudTrail permission to write to it. For more information, see Setting Bucket Policy (p. 34).
4. Click **Advanced**.
5. In the **Log file prefix** field, enter the same prefix you entered when you turned on CloudTrail in the first region.

Note
If you choose to use a prefix that is different from the one you entered when you turned on CloudTrail in the first region, you must edit the bucket policy on your destination bucket to allow CloudTrail to write log files to your bucket using this new prefix.

6. Select **No** for **Include global services?** to avoid recording duplicate API calls from global services such as IAM or AWS STS.

   Note
   Global services generate the same events in all regions, so selecting **Yes** for regions other than your first one results in duplicate entries for the global events in the log files.

7. (Optional) Select **Yes** or **No** for **SNS notification for every log file delivery?** If you select **Yes**, enter a name for your Amazon SNS topic in the **SNS topic (new)** field.

   Note
   Amazon SNS is a regional service, so if you choose to create a topic, that topic will exist in the same region in which you are enabling CloudTrail. If you have enabled CloudTrail's SNS notification in more than one region, you need to subscribe to CloudTrail's SNS topic in each region where one exists. See the Amazon Simple Notification Service Getting Started Guide for more information.

8. Click **Subscribe** if you are turning on CloudTrail for the first time or **Save if it is already turned on.**.
9. In about 15 minutes, CloudTrail starts publishing log files that show the AWS calls made in your accounts in this region since you completed the preceding steps.

Aggregating Log Files from Multiple Accounts

You can aggregate CloudTrail log files from multiple AWS accounts into a single Amazon S3 bucket. For example, you have four AWS accounts with account IDs 111111111111, 222222222222, 333333333333, and 444444444444, and you want to configure CloudTrail to deliver log files from all four of these accounts to a bucket belonging to account 111111111111. To accomplish this, complete the following steps in order:

1. Turn on CloudTrail in the account where the destination bucket will belong (111111111111 in this example). Do not turn on CloudTrail in any other accounts yet.

   For instructions, see Turning On CloudTrail for the First Time (p. 32).
2. Update the bucket policy on your destination bucket to grant cross-account permissions to CloudTrail.

   For instructions, see Setting Bucket Policy (p. 36).
3. Turn on CloudTrail in the other accounts you want (222222222222, 333333333333, and 444444444444 in this example). Configure CloudTrail in these accounts to use the same bucket belonging to the account that you specified in step 1 (111111111111 in this example).

   For instructions, see Turning on CloudTrail in Additional Accounts (p. 38).

Topics

Setting Bucket Policy

To enable cross-account log file aggregation, the bucket policy must grant CloudTrail permission to write log files from all the accounts you specify. This means that you must modify the bucket policy on your destination bucket to grant CloudTrail permission to write log files from each specified account.

To modify bucket permissions to allow aggregation from multiple accounts

1. Sign in to the AWS Management Console using the account that owns the bucket (111111111111 in this example) and open the Amazon S3 console.
2. Select the bucket where CloudTrail delivers your log files and then click **Properties**.
3. Click **Permissions**.
4. Click **Edit Bucket Policy**.
5. Modify the existing policy to add a line for each additional account whose log files you want delivered to this bucket. See the following example policy and note the underlined `Resource` line specifying a second account ID.

   Note
   An AWS account ID is a twelve-digit number, and leading zeros must not be omitted.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSCloudTrailAclCheck20131101",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::903692715234:root",
          "arn:aws:iam::859597730677:root",
          "arn:aws:iam::814480443879:root",
          "arn:aws:iam::216624486486:root",
          "arn:aws:iam::086441151436:root",
          "arn:aws:iam::388731089494:root",
          "arn:aws:iam::284668455005:root",
          "arn:aws:iam::113285607260:root",
          "arn:aws:iam::035351147821:root"
        ]
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::myBucketName"
    },
    {
      "Sid": "AWSCloudTrailWrite20131101",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::903692715234:root",
          "arn:aws:iam::859597730677:root",
          "arn:aws:iam::814480443879:root",
          "arn:aws:iam::216624486486:root",
          "arn:aws:iam::086441151436:root",
          "arn:aws:iam::388731089494:root",
          "arn:aws:iam::284668455005:root",
          "arn:aws:iam::113285607260:root",
          "arn:aws:iam::035351147821:root"
        ]
      },
      "Action": "s3:PutObject",
      "Resource": [
        "arn:aws:s3:::myBucketName/[optional] myLogFilePrefix/AWS
Logs/111111111111/*",
        "arn:aws:s3:::myBucketName/[optional] myLogFilePrefix/AWS
```

```
Logs/222222222222/*"
      ],
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}
```

Note
In the `Principal` fields, the account number must match the CloudTrail account for that region. For more information about the supported regions and their associated account numbers, see Supported Regions (p. ?).

Turning on CloudTrail in Additional Accounts

You can use the console or the command line interface to turn on CloudTrail in additional AWS accounts. For more information, see the following topics:

Topics

Using the Console to Turn on CloudTrail in Additional AWS Accounts

You can use the CloudTrail console to turn on CloudTrail in additional accounts.

1. Sign into the AWS management console using account 222222222222 credentials and open the AWS CloudTrail console. In the navigation bar, select the region where you want to turn on CloudTrail.
2. Click **Get Started** or, if you have already turned on CloudTrail, click **Edit**.
3. On the following page, for **Create a new S3 bucket?**, select **No**. Use the text box to enter the name of the bucket you created previously for storing log files when you signed in using account 111111111111 credentials. CloudTrail displays a warning asking you if you are sure that you want to specify an S3 bucket in another account. Verify the name of the bucket you entered, and if you are sure it's the correct one, click **Continue**.
4. Click **Advanced**.
5. In the **Log file prefix** field, enter the same prefix you entered for storing log files when you turned on CloudTrail using account 111111111111 credentials. If you choose to use a prefix that is different from the one you entered when you turned on CloudTrail in the first account, you must edit the bucket policy on your destination bucket to allow CloudTrail to write log files to your bucket using this new prefix.
6. Select **No** for **Include global services?** to avoid recording duplicate API calls from global services such as IAM or AWS STS.

   Note
   Global services generate the same events in all regions, so selecting **Yes** for regions other than your first one results in duplicate entries for the global events in the log files.
7. (Optional) Select **Yes** or **No** for **SNS notification for every log file delivery?**. If you select **Yes**, enter a name for your Amazon SNS topic in the **SNS topic (new) field**.

Note
Amazon SNS is a regional service, so if you choose to create a topic, that topic will exist in the same region in which you are enabling CloudTrail. If you have turned on CloudTrail SNS notification in more than one region, you need to subscribe to the CloudTrail SNS topic in each region. See the Amazon Simple Notification Service Getting Started Guide for more information.

8. Click **Subscribe** if you are turning on CloudTrail for the first time or **Save if it is already turned on.**.
9. In about 15 minutes, CloudTrail starts publishing log files that show the AWS calls made in your accounts for the selected region since you completed the preceding steps.

Using the CLI to Turn on CloudTrail in Additional AWS Accounts

You can use the AWS command line tools to turn on CloudTrail in additional accounts and aggregate their log files to one Amazon S3 bucket. For more information about these tools, see the AWS Command Line Interface User Guide.

Turn on CloudTrail in your additional accounts by using the `create-subscription` command. Use the following options to specify additional settings:

- `--name` specifies the name of the trail.
- `--s3-use-bucket` specifies the existing Amazon S3 bucket, created when you turned on CloudTrail in your first account (111111111111 in this example).
- `--s3-prefix` specifies a prefix for the log file delivery path (optional).
- `--sns-new-topic` specifies the name of the Amazon SNS topic to which you can subscribe for notification of log file delivery to your bucket (optional).

In contrast to trails that you create using the console, you must give every trail you create with the AWS CLI a name. You can create one trail for each region in which an account is running AWS resources.

The following example command shows how to create a trail for your additional accounts by using the AWS CLI. To have log files for these account delivered to the bucket you created in your first account (111111111111 in this example), specify the bucket name in the `--s3-new-bucket` option. Amazon S3 bucket names are globally unique.

```
aws cloudtrail create-subscription --name AWSCloudTrailExample --s3-use-bucket
 MyBucketBelongingToAccount111111111111 --s3-prefix AWSCloudTrailPrefixExample
 --sns-new-topic AWSCloudTrailLogDeliveryTopicExample
```

When you run the command, you will see output similar to the following:

```
CloudTrail configuration:
{
  "trailList": [
    {
      "S3KeyPrefix": "AWSCloudTrailPrefixExample",
      "IncludeGlobalServiceEvents": true,
      "Name": "AWSCloudTrailExample",
      "SnsTopicName": "AWSCloudTrailLogDeliveryTopicExample",
      "S3BucketName": "MyBucketBelongingToAccount111111111111"
    }
  ]
}
```

For more information about using CloudTrail from the AWS command line tools, see the AWS CloudTrail Command Line Reference.

Sharing CloudTrail Log Files Between AWS Accounts

This section explains how to share CloudTrail log files between multiple AWS accounts. We will assume that the log files have all been aggregated into a single Amazon S3 bucket. You do not need to aggregate log files in order to share them, but aggregating log files across accounts and regions is quite common and we will assume that you have done so when discussing the two scenarios that follow. In the first scenario, you will learn how to grant read-only access to the accounts that generated the log files that have been aggregated into your Amazon S3 bucket. In the second scenario, you will learn how to grant access to all of the log files to a third-party account that can analyze the files for you.

To share log files between multiple AWS accounts, you must perform the following general steps. These steps are explained in detail later in this section.

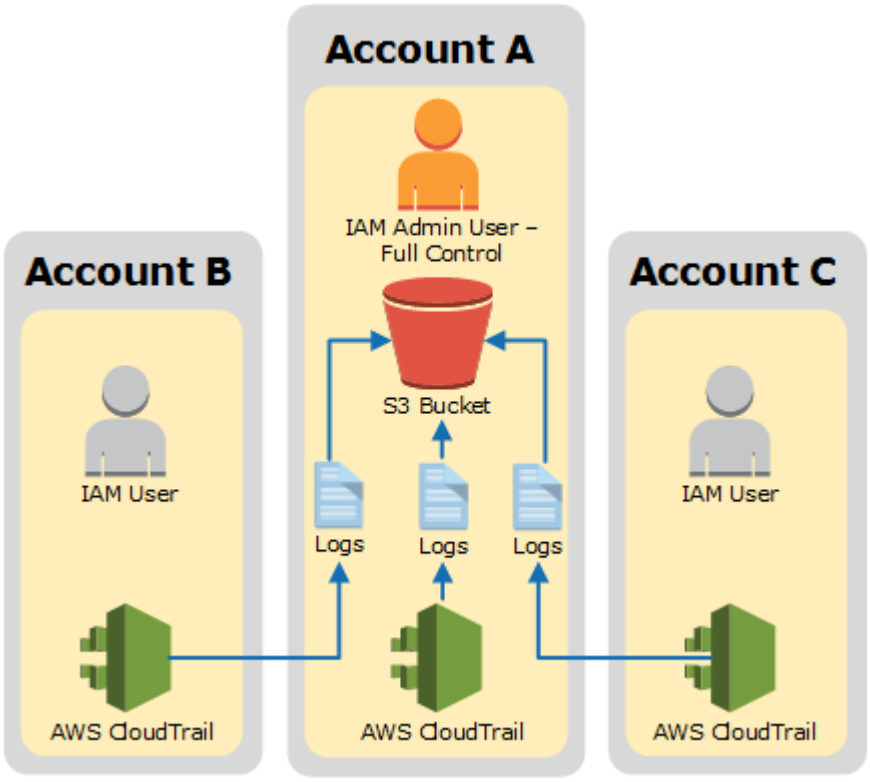
- Create an IAM role for each account that you want to share log files with.
- For each of these IAM roles, create an access policy that grants read-only access to the account you want to share the log files with.
- Have an IAM user in each account programmatically assume the appropriate role and retrieve the log files.

This section walks you through the preceding steps in the context of two different sharing scenarios: granting access to the log files to each account that generated those files, and sharing log files with a third party. Most of the steps you take for the two scenarios are the same; the important difference is in what kind of permissions the IAM role grants to each account. That is, you can grant permission for an account to read only its own log files, or you can grant an account permission to read all log files. For details about permissions management for IAM roles, see Roles (Delegation and Federation) in *Using IAM*.

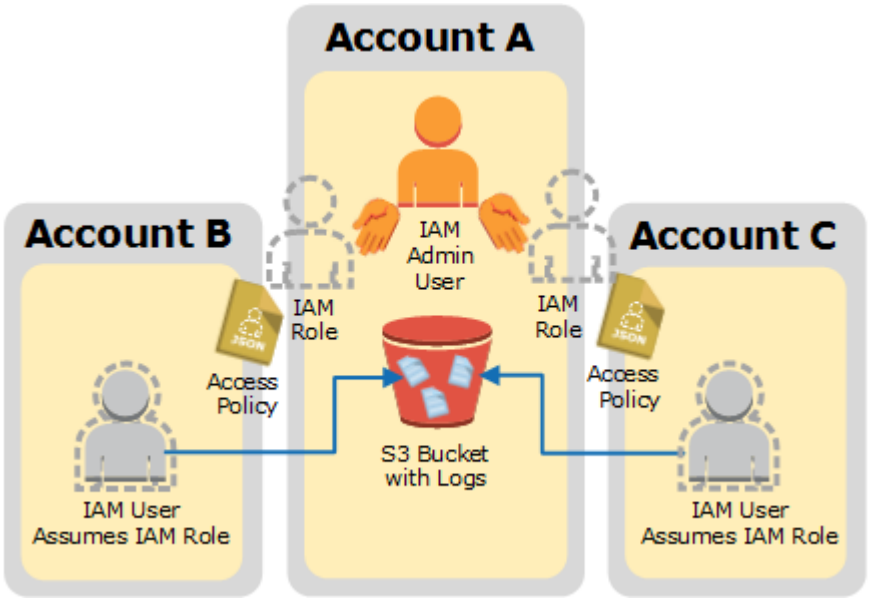
Scenario 1: Granting Access to the Account that Generated the Log Files

In this scenario, we'll assume that your enterprise is made up of two business units and that it maintains three AWS accounts. The first account, Account A, is the top-level account. For example, it might be managed by your enterprise's IT department and therefore be responsible for aggregating log files from all other departments and business units. The other two accounts, B and C, correspond to your enterprise's business units.

This scenario assumes that you have already aggregated the log files from all three accounts into a single Amazon S3 bucket, and that account A has full control over that bucket, as shown in the following illustration.



Although the Amazon S3 bucket contains log files that were generated by Accounts A, B and C, accounts B and C do not initially have access to the log files that accounts B and C generated. You will give each business unit read-only access to the log files that it generated, as shown in the following illustration.



To grant read-only access to the log files generated by accounts B and C, you must do the following in the account Account A. Remember that Account A has full control Amazon S3 bucket.

- Create an IAM role for account B and another IAM role for account C. How: Creating a Role (p. 43)

- For the IAM role created for account B, create an access policy that grants read-only access to the log files generated by account B. For the IAM role created for account C, create an access policy that grants read-only access to the log files generated by account C. How: Creating an Access Policy to Grant Access to Accounts You Own (p. 44)
- Have an IAM user in account B programmatically assume the role created for account B. Have an IAM user in account C programmatically assume the role created for account C. Each IAM user must be given permission to assume the role by the respective account owner. How: Creating permissions policies for IAM users (p. 48).
- Finally, the account owner who grants the permission must be an administrator, and must know the ARN of the role in account A that is being assumed. How: Calling AssumeRole (p. 48).

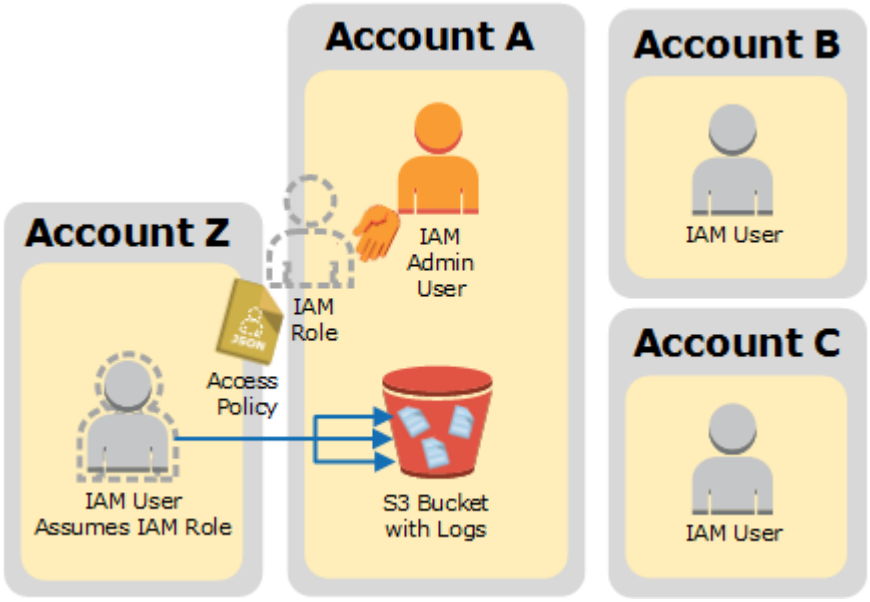
The IAM users in accounts B and C can then programmatically retrieve their own log files, but not the log files of any other account.

Scenario 2: Granting Access to All Logs

In this scenario, we'll assume that your enterprise is structured as it was in the previous scenario, that is, it is made up of two business units and it maintains three AWS accounts. The first account, Account A, is the top-level account. For example, it might be managed by your enterprise's IT department and therefore be responsible for aggregating all other log files. The other two accounts, B and C, correspond to each of your enterprise's business units.

Like the previous scenario, this scenario assumes that you have already aggregated the log files from all three accounts into a single Amazon S3 bucket, and that account A has full control over that bucket.

Finally, we'll also assume that your enterprise wants to share all the log files from all accounts (A, B, and C) with a third party. We'll say that the third party has an AWS account called Account Z, as shown in the following illustration.



To share all of the log files from your enterprise with Account Z, you must do the following in the Account A, the account that has full control over the Amazon S3 bucket.

- Create an IAM role for Account Z. How: Creating a Role (p. 43)

- For the IAM role created for Account Z, create an access policy that grants read-only access to the log files generated by accounts A, B, and C. How: Creating an Access Policy to Grant Access to a Third Party (p. 46)
- Have an IAM user in Account Z programmatically assume the role and then retrieve the appropriate log files. The IAM user must be given permission to assume the role by the owner of Account Z. How: Creating permissions policies for IAM users (p. 48). Further, the account owner who grants the permission must be an administrator and know the ARN of the role in Account A that is being assumed. How: Calling AssumeRole (p. 48).

Topics

Creating a Role

When you aggregate log files from multiple accounts into a single Amazon S3 bucket, only the account that has full control of the bucket, Account A in our example, has full read access to all of the log files in the bucket. Accounts B, C, and Z in our example do not have any rights until granted. Therefore, to share your AWS CloudTrail log files from one account to another (that is, to complete either Scenario 1 or Scenario 2 described previously in this section), you must *enable cross-account access*. You can do this by creating IAM roles and their associated access policies.

Roles

Create an IAM *role* for each account to which you want to give access. In our example, you will have three roles, one each for accounts B, C, and Z. Each IAM role defines an access or permissions policy that enables the accounts to access the resources (log files) owned by account A. The permissions are attached to each role and are associated with each account (B, C, or Z) only when the role is assumed. For details about permissions management for IAM roles, see Roles (Delegation and Federation) . For more information about how to assume a role, see Assuming a Role (p. 47).

Policies

There are two policies for each IAM role you create. The *trust policy* specifies a *trusted entity* or *principal*. In our example, accounts B, C, and Z are trusted entities, and an IAM user with the proper permissions in those accounts can assume the role.

The *trust policy* is automatically created when you use the console to create the role. If you use the SDK to create the role, you must supply the trust policy as a parameter to the `CreateRole` API. If you use the CLI to create the role, you must specify the trust policy in the `create-role` CLI command.

The *role access (or permissions) policy* that you must create as the owner of Account A defines what actions and resources the principal or trusted entity is allowed access to (in this case, the CloudTrail log files). For Scenario 1 that grants log file access to the account that generated the log files, as discussed in Creating an Access Policy to Grant Access to Accounts You Own (p. 44). For Scenario 2 that grants read access to all log files to a third party, as discussed in Creating an Access Policy to Grant Access to a Third Party (p. 46).

For further details about creating and working with IAM policies, see Permissions and Policies in *Using IAM*.

Creating a Role

To Create a Role by Using the Console

1. Sign into the AWS Management Console as an administrator of Account A.
2. Navigate to the IAM console.
3. In the navigation pane, click **Roles**.
4. Click **Create New Role**.
5. Enter a name for the new role, and then click **Continue**.
6. Click **Roles for Cross-Account Access**.
7. For Scenario 1, do the following to provide access between accounts you own:
   a. Select **Provide access between AWS accounts you own.**
   b. Enter the twelve-digit account ID of the account (B, C, or Z) to be granted access.
   c. Check the **Require MFA** box if you want the user to provide multi-factor authentication before assuming the role.

   For Scenario 2, do the following to provide access to a third-party account. In our example, you would perform these steps for Account Z, the third-party log analyzer:
   a. Select **Allows IAM users from a 3rd party AWS account to access this account.**
   b. Enter the twelve-digit account ID of the account (Account Z) to be granted access.
   c. (Optional) Enter an external ID that can be used to provide additional control over who can assume the role. For more information, see About the External ID in the AWS Security Token Service User Guide .
8. Click **Continue** to set the permissions that you want to associate with this role.
9. Choose **Select Policy Template**.
10. Select the **Amazon S3 Read Only Access** template.

    Note
    By default, the template policy grants retrieval and list rights to all Amazon S3 buckets within your account.
    - To grant an account access to only that account's log files (Scenario 1), see Creating an Access Policy to Grant Access to Accounts You Own (p. 44).
    - To grant an account access to all of the log files in the Amazon S3 bucket (Scenario 2), see Creating an Access Policy to Grant Access to a Third Party (p. 46).
11. (Optional) Change the policy name.
12. Click **Continue** to review the role before you create it.
13. Click **Create Role**.

Creating an Access Policy to Grant Access to Accounts You Own

In Scenario 1, as an administrative user in Account A, you have full control over the Amazon S3 bucket to which CloudTrail writes log files for accounts B and C. You want to share each business unit's log files back to business unit that created them. But, you don't want a unit to be able to read any other unit's log files.

For example, to share Account B's log files with Account B but not with Account C, you must create a new IAM role in Account A that specifies that Account B is a trusted account. This role trust policy specifies that Account B is trusted to assume the role created by Account A, and should look like the following example. The trust policy is automatically created if you create the role by using the console. If you use the

SDK to create the role, you must supply the trust policy as a parameter to the `CreateRole` API. If you use the CLI to create the role, you must specify the trust policy in the `create-role` CLI command.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-B-id:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

You must also create an access policy to specify that Account B can read from only the location to which B wrote its log files. The access policy will look something like the following. Note that the **Resource** ARN includes the twelve-digit account ID for Account B, and the prefix you specified, if any, when you turned on CloudTrail for Account B during the aggregation process. For more information about specifying a prefix, see Turning on CloudTrail in Additional Accounts (p. 38).

Important

You must ensure that the prefix in the access policy is exactly the same as the prefix that you specified when you turned on CloudTrail for Account B. If it is not, then you must edit the IAM role access policy in Account A to incorporate the actual prefix for Account B. If the prefix in the role access policy is not exactly the same as the prefix you specified when you turned on CloudTrail in Account B, then Account B will not be able to access its log files.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/account-B-id/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::bucket-name"
    }
  ]
}
```

The role you create for Account C will be nearly identical to the one you created for Account B. The access policy for each role must include the appropriate account ID and prefix so that each account can read from only the location to which CloudTrail wrote that account's log files.

After you have created roles for each account and specified the appropriate trust and access policies, and after an IAM user in each account has been granted access by the administrator of that account, an IAM user in accounts B or C can programmatically assume the role.

After you have created roles for each account and specified the appropriate trust and access policies, an IAM user in one of the newly trusted accounts (B or C) must programmatically assume the role in order to read log files from the Amazon S3 bucket.

For more information, see Assuming a Role (p. 47).

Creating an Access Policy to Grant Access to a Third Party

Account A must create a separate IAM role for Account Z, the third-party analyzer in Scenario 2. The trust relationship, automatically created by AWS when you create the role, specifies that Account Z will be trusted to assume the role. The access policy for the role specifies what actions Account Z can take. For more information about creating roles and role policies, see Creating a Role (p. 43).

For example, the trust relationship created by AWS will specify that Account Z is trusted to assume the role created by Account A, and will look something like the following.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-Z-id:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

If you specified an external ID when you created the role for Account Z, your access policy contains an added **Condition** element that tests the unique ID assigned by Account Z. The test is performed when the role is assumed. The **Condition** element is shown in the following example access policy. For more information, see About the External ID .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-Z-id:root"
      },
```

```
      "Action": "sts:AssumeRole"
      "Condition": {
           "StringEquals": {
              "sts:ExternalId": "external ID-issued-by-account-Z"
            }
      }
    }
  ]
}
```

You must also create an access policy for the Account A role to specify that Account Z can read all logs from the Amazon S3 bucket. The access policy should look something like the following. The wild card (*) at the end of the **Resource** value indicates that Account Z can access any log file in the Amazon S3 bucket to which it has been granted access.

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

After you have created a role for Account Z and specified the appropriate trust relationship and access policy, an IAM user in Account Z must programmatically assume the role to be able to read log files from the Amazon S3 bucket. For more information, see Assuming a Role (p. 47).

Assuming a Role

You must designate a separate IAM user to assume each role you've created in each account, and ensure that each IAM user has appropriate permissions.

IAM Users and Roles

After you have created the necessary roles and policies in Account A for scenarios 1 and 2, you must designate an IAM user in each of the accounts B, C, and Z. Each IAM user will programmatically assume the appropriate role to access the log files. That is, the user in account B will assume the role created for account B, the user in account C will assume the role created for account C, and the user in account Z will assume the role created for account Z. When a user assumes a role, AWS returns temporary security

credentials that can be used to make requests to list, retrieve, copy, or delete the log files depending on the permissions granted by the access policy associated with the role.

For more information about working with IAM users, see Working with IAM Users and Groups .

The primary difference between scenarios 1 and 2 is in the access policy that you create for each IAM role in each scenario.

- In scenario 1, the access policies for accounts B and C limit each account to reading only its own log files. For more information, see Creating an Access Policy to Grant Access to Accounts You Own (p. 44).
- In scenario 2, the access policy for Account Z allows it to read all the log files that are aggregated in the Amazon S3 bucket. For more information, see Creating an Access Policy to Grant Access to a Third Party (p. 46).

Creating permissions policies for IAM users

To perform the actions permitted by the roles, the IAM user must have permission to call the AWS STS `AssumeRole` API. You must edit the *user-based policy* for each IAM user to grant them the appropriate permissions. That is, you set a **Resource** element in the policy that is attached to the IAM user. The following example shows a policy for an IAM user in Account B that allows the user to assume a role named "Test" created earlier by Account A.

To attach the required policy to the IAM role

1. Sign in to the AWS Management Console and open the IAM console.
2. Select the user whose permissions you want to modify.
3. Select the **Permissions** tab.
4. Select **Custom Policy**.
5. Select **Use the policy editor to customize your own set of permissions**.
6. Give the policy a name.
7. Copy the following policy into the space provided for the policy document.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole"],
      "Resource": "arn:aws:iam::account-A-id:role/Test"
    }
  ]
}
```

Important
Only IAM users can assume a role. If you attempt to use AWS root account credentials to assume a role, access will be denied.

Calling AssumeRole

A user in accounts B, C, or Z can assume a role by creating an application that calls the AWS STS `AssumeRole` API and passes the role session name, the Amazon Resource Number (ARN) of the role to assume, and an optional external ID. The role session name is defined by Account A when it creates the role to assume. The external ID, if any, is defined by Account Z and passed to Account A for inclusion

during role creation. For more information, see About the External ID . You can retrieve the ARN from the Account A by opening the IAM console.

To Find the ARN Value in Account A

- Select **Roles**
- Select the role you want to examine.
- Look for the Role ARN in the **Summary** tab

The AssumeRole API returns temporary credentials that a user in accounts B, C, or Z can use to access resources in Account A. In this example, the resources you want to access are the Amazon S3 bucket and the log files that the bucket contains. The temporary credentials have the permissions that you defined in the role access policy.

The following Python example (using the AWS SDK for Python (Boto)) shows how to call `AssumeRole` and how to use the temporary security credentials returned to list all Amazon S3 buckets controlled by Account A.

```
import boto
from boto.sts import STSConnection
from boto.s3.connection import S3Connection

# The calls to AWS STS AssumeRole must be signed using the access key ID and
secret
# access key of an IAM user or using existing temporary credentials. (You cannot
 call
# AssumeRole using the access key for an account.) The credentials can be in
# environment variables or in a configuration file and will be discovered
automatically
# by the STSConnection() function. For more information, see the Python SDK
# documentation: http://boto.readthedocs.org/en/latest/boto_config_tut.html

sts_connection = STSConnection()
assumedRoleObject = sts_connection.assume_role(
    role_arn="arn:aws:iam::account-of-role-to-assume:role/name-of-role",
    role_session_name="AssumeRoleSession1"
)

# Use the temporary credentials returned by AssumeRole to call Amazon S3
# and list the bucket in the account that owns the role (the trusting account)
s3_connection = S3Connection(
    aws_access_key_id=assumedRoleObject.credentials.access_key,
    aws_secret_access_key=assumedRoleObject.credentials.secret_key,
    security_token=assumedRoleObject.credentials.session_token
)
bucket = s3_connection.get_bucket(bucketname)
print bucket.name
```

Stop Sharing

To stop sharing log files to another AWS account, simply delete the role that you created for that account in Creating a Role (p. 43).

1. Sign in to the AWS Management Console as an IAM user with administrative-level permissions for Account A.

2. Navigate to the IAM console.
3. In the navigation pane, click **Roles**.
4. Select the role you want to delete.
5. Right-click and select **Delete Role** from the context menu.

Monitoring CloudTrail Log Files with Amazon CloudWatch Logs

Topics

One of the ways that you can work with CloudTrail logs is to monitor them in real time by sending them to CloudWatch Logs. You can configure CloudTrail to send log files to a CloudWatch Logs log group. You define CloudWatch Logs metric filters that will evaluate your CloudTrail log events for matches in terms, phrases, or values. You assign CloudWatch metrics to the metric filters. You also create CloudWatch alarms that are triggered according to thresholds and time periods that you specify. You can configure an alarm to send a notificationwhen the alarm is triggered so that you can take immediate action. You can also configure CloudWatch to automatically perform an action in response to an alarm. CloudTrail events are protected by SSL encryption as they are delivered from CloudTrail to the CloudWatch Logs log group.

CloudWatch Logs is supported in the `us-east-1`, `us-west-2`, and `eu-west-1` regions. Standard pricing for Amazon CloudWatch and Amazon CloudWatch Logs apply. For more information, see Amazon CloudWatch Pricing.

Sending CloudTrail Events to CloudWatch Logs

This topic describes how to configure CloudTrail to send logs to CloudWatch Logs so that you can monitor CloudTrail log events. Sending CloudTrail events to a CloudWatch Logs log group requires you to do the following:

- Create a log group.
- Create an IAM role.
- Attach a role policy.
- Assume the role.
- Update the trail to support CloudWatch Logs as a delivery endpoint.

> **Note**
> You must create a trail before you can configure CloudTrail to send log events to CloudWatch Logs. The procedures in this section assume you have already created a trail. For more information, see Creating and Updating a Trail with the CloudTrail Console (p. 20) or Creating and Updating a Trail with the AWS CLI (p. 21). Also make sure to configure permissions for your IAM user to create a log group, change the log group, and assume the role by creating an action policy. For more information, see Granting Custom Permissions (p. 25).

This topic includes procedures for the AWS Management Console and the AWS Command Line Tool (CLI).

Configuring CloudWatch Logs Monitoring Using the Console

You can use the AWS Management to configure CloudTrail to send log events to CloudWatch Logs for monitoring.

Creating a Log Group or Specifying an Existing Log Group

CloudTrail uses a CloudWatch Logs log group as a delivery endpoint for log events. You can create a new log group or specify an existing one.

To specify a log group using the console

1. Navigate to the **CloudTrail Configuration** page.
2. Next to **CloudWatch Logs (Optional)**, click the **Edit** (pencil) icon.
3. In the **New or existing log group** box, type a log group name to organize CloudTrail events for you to review using CloudWatch Logs, and then click **Continue**.

Next, create a role for CloudTrail to assume to deliver events to the log stream.

Creating a Role

To create a role using the console

1. Click **View Details**. Then, next to **IAM Role**, verify that **Create a new IAM role** is selected.

   Note
   You can specify an existing role, but you must attach the appropriate role policy to the existing role if you want to use it to send log events to CloudWatch Logs.
2. Optionally, click **Role Name** and then type a role name.

Attaching a Role Policy

This section is continuation of the previous section.

To attach a role policy

1. Click **View Policy Document**. If you created a new role in the previous step, the policy document will contain the following required permissions to create a CloudWatch Logs log stream in the log group that you specify and to deliver CloudTrail events to that log stream. (This is the default policy for the default IAM role `CloudTrail_CloudWatchLogs_Role`.)

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

      "Sid": "AWSCloudTrailCreateLogStream2014110",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": [
```

```
        "arn:aws:logs:us-east-1:accountID:log-group:log_group_name:log-
stream:CloudTrail_log_stream_name_prefix*"
      ]

    },
    {
      "Sid": "AWSCloudTrailPutLogEvents20141101",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:accountID:log-group:log_group_name:log-
stream:CloudTrail_log_stream_name_prefix*"
      ]
    }
  ]
}
```

2. When you are finished, click **Allow**.

When you are finished with these steps in the console, the CloudTrail trail will be set up to use the log group and role you created to send events to CloudWatch Logs.

Configuring CloudWatch Logs Monitoring Using the CLI

You can use the AWS CLI to configure CloudTrail to send log events to CloudWatch Logs for monitoring.

Creating a Log Group

1. If you do not already have an existing log group, create a CloudWatch Logs log group as a delivery endpoint for log events using the CloudWatch Logs command `create-log-group`.

   ```
   aws logs create-log-group --log-group-name name
   ```

   For example:

   ```
   aws logs create-log-group --log-group-name CloudTrail/logs
   ```

2. Retrieve the log group Amazon Resource Name (ARN).

   ```
   aws logs describe-log-groups
   ```

Create a Policy Document

When you use the CLI to create a role and attach a role policy, you must create a policy document as a JSON file. Open a text editor. Type the following and then save the policy document with the `.json` file extension. The following grants CloudTrail the required permissions to create a CloudWatch Logs log stream in the log group that you specify and to deliver CloudTrail events to that log stream. (This is the default policy for the default IAM role `CloudTrail_CloudWatchLogs_Role`.)

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

      "Sid": "AWSCloudTrailCreateLogStream2014110",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:accountID:log-group:log_group_name:log-
stream:CloudTrail_log_stream_name_prefix*"
      ]

    },
    {
      "Sid": "AWSCloudTrailPutLogEvents20141101",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:accountID:log-group:log_group_name:log-
stream:CloudTrail_log_stream_name_prefix*"
      ]
    }
  ]
}
```

Creating a Role

Create a role for CloudTrail to use to send events to the CloudWatch Logs log group using the IAM `create-role` command.

```
aws iam create-role --role-name role_name --assume-role-policy-document
file://policy_document.json
```

Running this command attaches a trust policy to the specified role so that CloudTrail can assume the role. Take note of the role ARN.

Updating the Trail

Update the trail with the log group and role information using the CloudTrail `update-trail` command.

```
aws cloudtrail update-trail --name trail_name --cloud-watch-logs-log-group-arn
 log_group_arn --cloud-watch-logs-role_arn role_arn
```

For more information about the AWS CLI commands in these procedures, see the AWS CloudTrail Command Line Reference.

Creating CloudWatch Alarms for CloudTrail Events

This section describes how to configure alarms for CloudTrail events using example scenarios. Configuring alarms involves the following steps:

- Create a metric filter.
- Create an alarm.
- Create a notification or autoscaling action for when the alarm is triggered.

Topics

Example: Console Sign-In Failures

This scenario walks you through how to use the AWS Management Console to create an Amazon CloudWatch alarm that is triggered when there are three or more sign-in failures during a five minute period.

Create a Metric Filter

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, click **Logs**.
3. In the list of log groups, select the check box next to the log group that you created for CloudTrail log events.
4. Click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, click **Filter Pattern** and then type the following:

```
{ ($.eventName = "ConsoleLogin") && ($.errorMessage = "Failed authentication")
 }
```

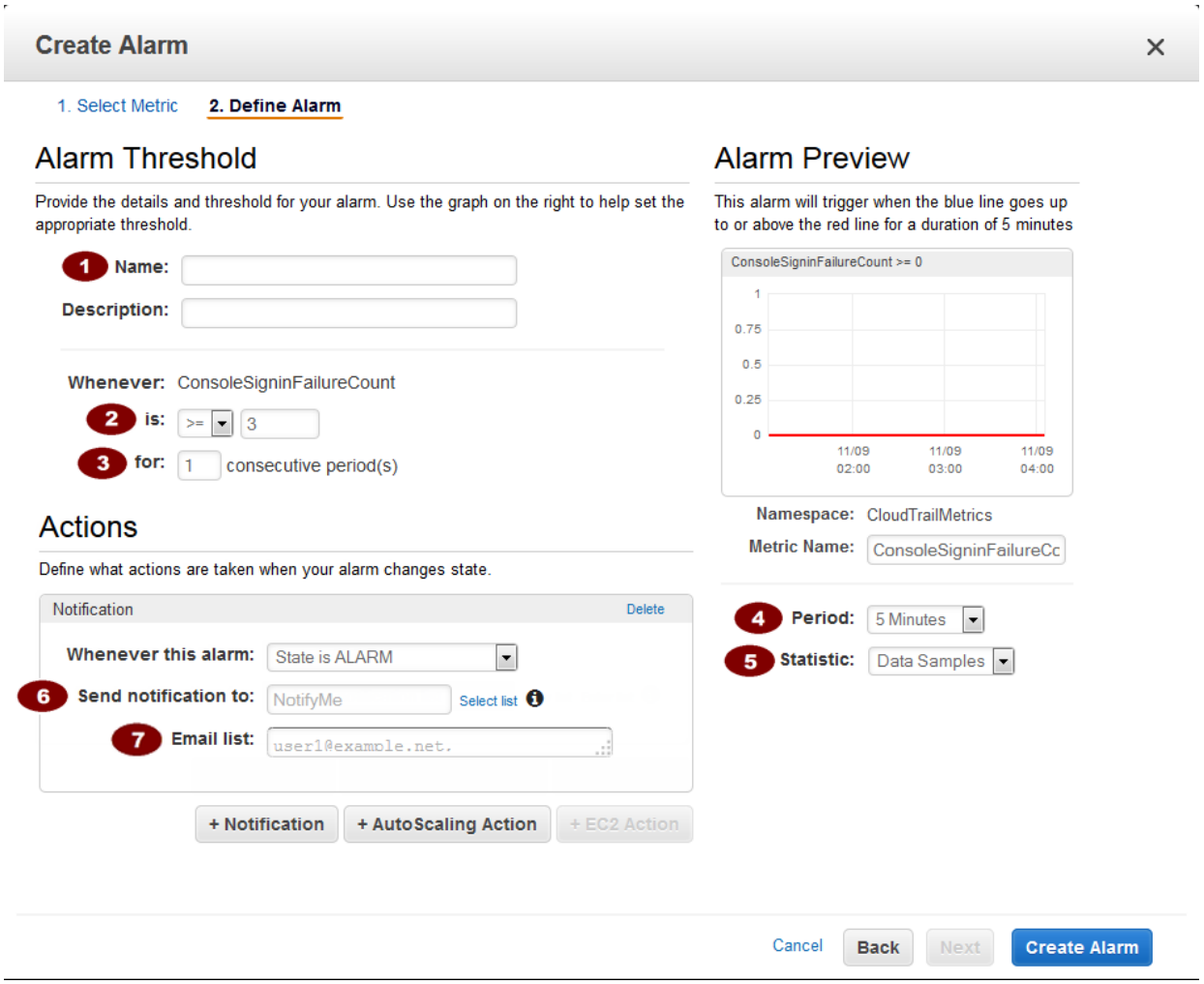
   Note
   For more information about syntax for metric filters and patterns for CloudTrail log events, see the JSON-related sections of Filter and Pattern Syntax in the Amazon CloudWatch User Guide.

6. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** box, enter **ConsoleSignInFailures**
7. Under **Metric Details**, in the **Metric Namespace** box, enter **CloudTrailMetrics**.
8. In the **Metric Name** box, enter **ConsoleSigninFailureCount**.
9. Click **Show advanced metric settings**.
10. Click **Metric Value**, and then type **1**.
11. When you are finished, click **Create Filter**.

Creating an Alarm

These steps are a continuation of the previous steps for creating a metric filter.

1. On the **Filters for *Log_Group_Name*** page, next to the filter name, click **Create Alarm**.
2. On the **Create Alarm** page, provide the following values.



Setting	Value
1	Console Sign-in Failures Alarm
2	>=3
3	1
4	5 Minutes
5	Data Samples
6	Click **New list**, and then click **Send notification to** and type a unique name for this notification list.
7	Click **Email list**, and then type the email address to which you want notifications sent. (You will receive an email at this address to confirm that you created this alarm.)

3. When you are finished, click **Create Alarm**.

Example: Network Access Control List (ACL) Changes

This scenario walks you through how to use the AWS Management Console to create an Amazon CloudWatch alarm that is triggered when any configuration changes happen involving network ACLs.

Create a Metric Filter

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, click **Logs**.
3. In the list of log groups, select the check box next to the log group that you created for CloudTrail log events.
4. Click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, click **Filter Pattern** and then type the following:

```
{ ($.eventName = "CreateNetworkAcl") || ($.eventName = "CreateNetwork
AclEntry") || ($.eventName = "DeleteNetworkAcl") || ($.eventName = "DeleteN
etworkAclEntry") || ($.eventName = "ReplaceNetworkAclEntry") || ($.eventName
 = "ReplaceNetworkAclAssociation") }
```

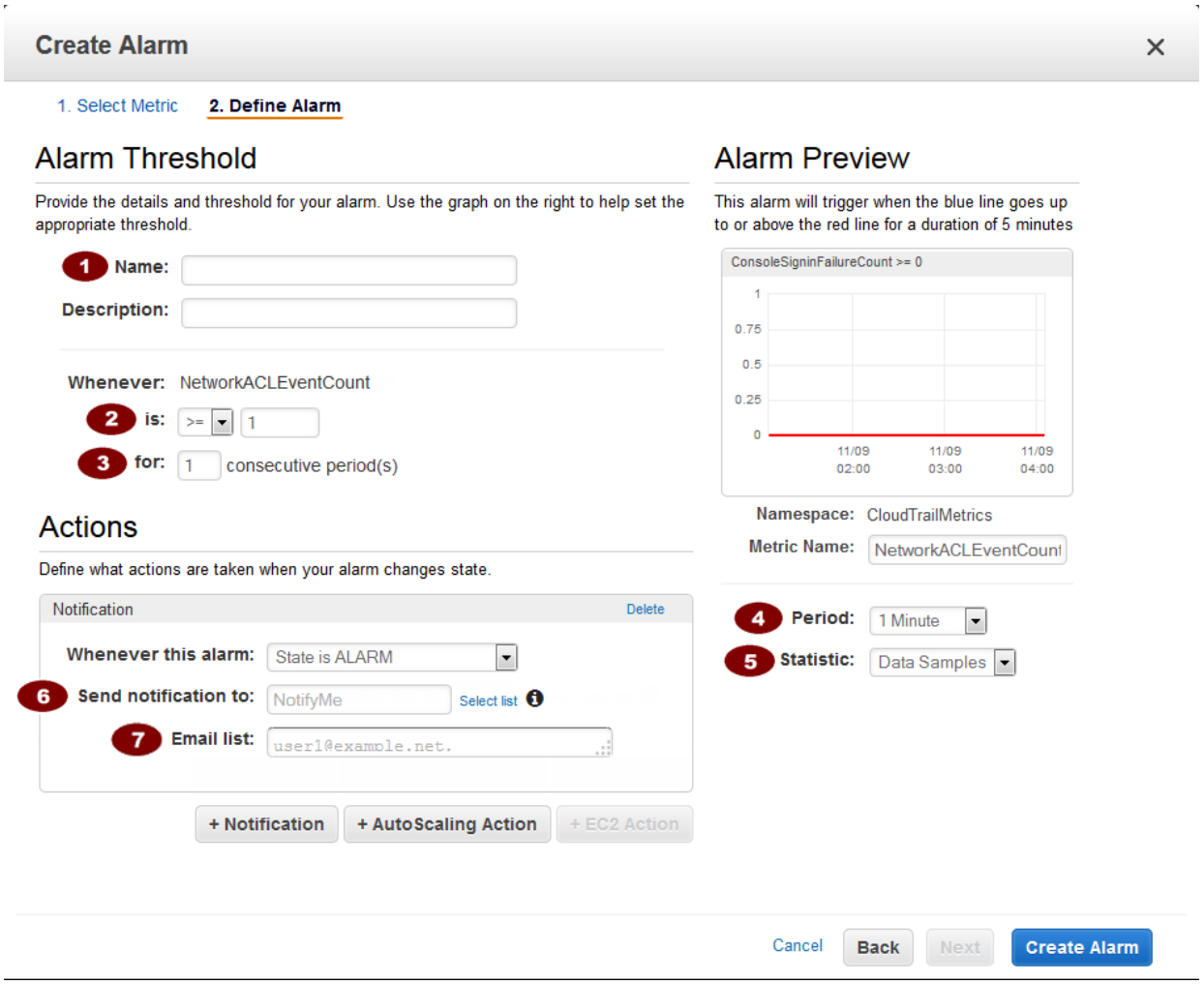
   Note
   For more information about syntax for metric filters and patterns for CloudTrail log events, see the JSON-related sections of Filter and Pattern Syntax in the Amazon CloudWatch User Guide.

6. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** box, enter **NetworkACLEvents**.
7. Under **Metric Details**, in the **Metric Namespace** box, enter **CloudTrailMetrics**.
8. In the **Metric Name** box, enter **NetworkACLEventCount**.
9. Click **Show advanced metric settings**.
10. Click **Metric Value**, and then type **1**.
11. When you are finished, click **Create Filter**.

Creating an Alarm

These steps are a continuation of the previous steps for creating a metric filter.

1. On the **Filters for *Log_Group_Name*** page, next to the filter name, click **Create Alarm**.
2. On the **Create Alarm** page, provide the following values.



Setting	Value
1	Network ACL Configuration Changes Alarm
2	>=1
3	1
4	1 Minute
5	Data Samples
6	Click **New list**, and then click **Send notification to** and type a unique name for this notification list.
7	Click **Email list**, and then type the email address to which you want notifications sent. (You will receive an email at this address to confirm that you created this alarm.)

3. When you are finished, click **Create Alarm**.

Example: Security Group Configuration Changes

This scenario walks you through how to use the AWS Management Console to create an Amazon CloudWatch alarm that is triggered when any configuration changes happen involving security groups.

Create a Metric Filter

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, click **Logs**.
3. In the list of log groups, select the check box next to the log group that you created for CloudTrail log events.
4. Click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, click **Filter Pattern** and then type the following:

```
{ ($.eventName = "AuthorizeSecurityGroupIngress") || ($.eventName = "Author
izeSecurityGroupEgress") || ($.eventName = "RevokeSecurityGroupIngress" ||
 ($.eventName = "RevokeSecurityGroupEgress") || ($.eventName = "CreateSecur
ityGroup") || ($.eventName = "DeleteSecurityGroup") }
```

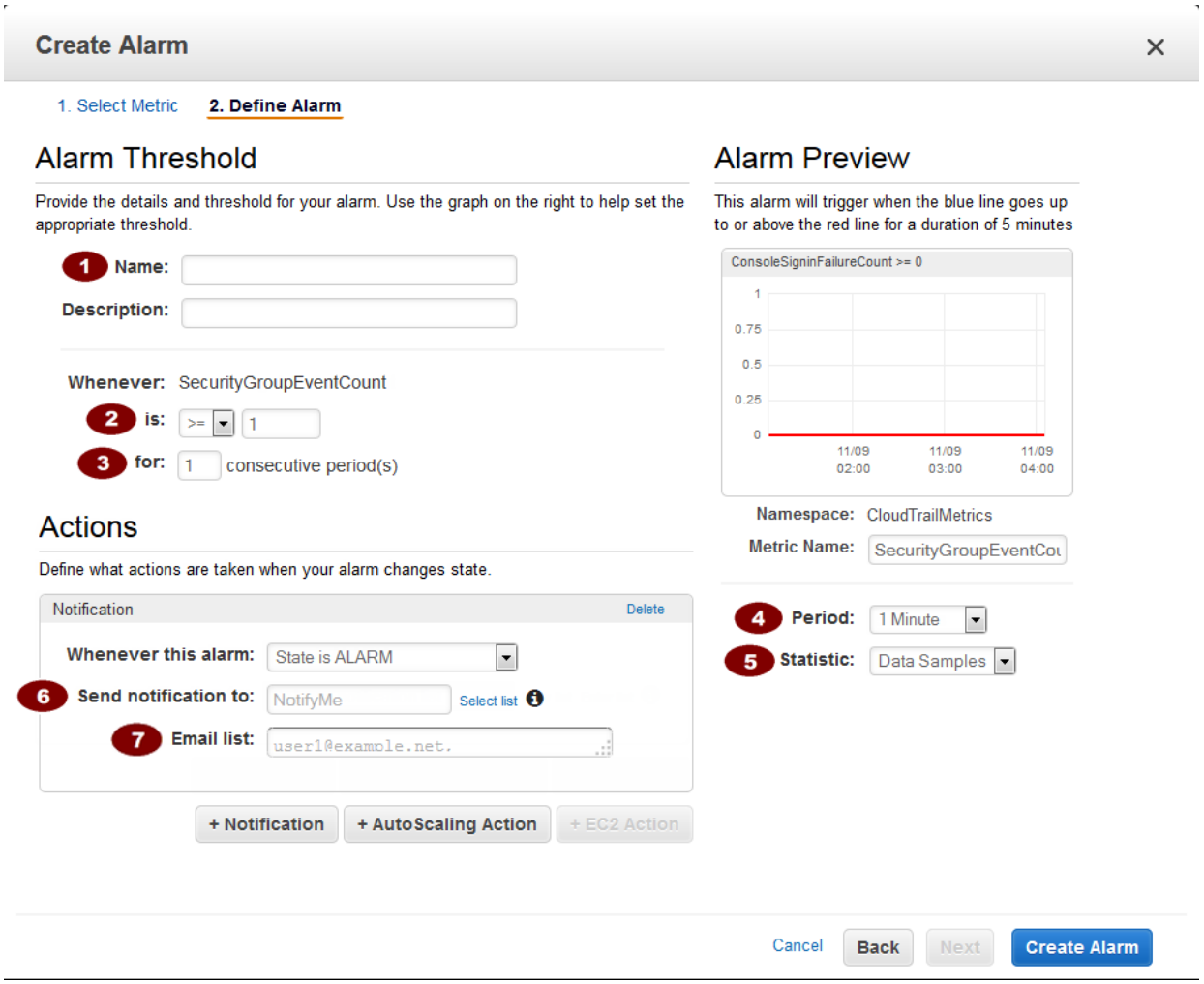
   Note
   For more information about syntax for metric filters and patterns for CloudTrail log events, see the JSON-related sections of Filter and Pattern Syntax in the Amazon CloudWatch User Guide.

6. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** box, enter **SecurityGroupEvents**
7. Under **Metric Details**, in the **Metric Namespace** box, enter **CloudTrailMetrics**.
8. In the **Metric Name** field, enter **SecurityGroupEventCount**.
9. Click **Show advanced metric settings**.
10. Click **Metric Value**, and then type **1**.
11. When you are finished, click **Create Filter**.

Creating an Alarm

These steps are a continuation of the previous steps for creating a metric filter.

1. On the **Filters for *Log_Group_Name*** page, next to the filter name, click **Create Alarm**.
2. On the **Create Alarm** page, provide the following values.



Setting	Value
1	Security Group Configuration Changes Alarm
2	>=1
3	1
4	1 Minute
5	Data Samples
6	Click **New list**, and then click **Send notification to** and type a unique name for this notification list.
7	Click **Email list**, and then type the email address to which you want notifications sent. (You will receive an email at this address to confirm that you created this alarm.)

3. When you are finished, click **Create Alarm**.

Configuring Notifications for CloudWatch Logs Alarms

You can configure CloudWatch Logs to send a notification whenever an alarm is triggered for CloudTrail. Doing so enables you to respond quickly to critical operational events captured in CloudTrail events and detected by CloudWatch Logs. CloudWatch uses Amazon Simple Notification Service (SNS) to send email. For more information, see Set Up Amazon SNS in the CloudWatch Developer Guide.

Stopping CloudTrail from Sending Events to CloudWatch Logs

You can stop sending events to CloudWatch Logs by deleting the delivery endpoint.

AWS Management Console

To remove the CloudWatch Logs delivery endpoint using the AWS Management Console

1. Sign in to the AWS Management Console.
2. Navigate to the CloudTrail console.
3. In the navigation pane, click **Configuration**.
4. In the **CloudWatch Logs (optional)** section, click the **Delete** (trash can) icon.
5. Click **Continue** to confirm.

AWS Command Line Interface (CLI)

You can remove the CloudWatch Logs log group as a delivery endpoint using the `update-trail` command. The following command clears the log group and role from the trail configuration.

```
aws cloudtrail update-trail --name trailname --cloud-watch-logs-log-group-arn=""
 --cloud-watch-logs-role-arn=""
```

Log Group and Log Stream Names

Amazon CloudWatch will display the log group that you created for CloudTrail events alongside any other log groups you have in a region. We recommend that you use a log group name that helps you easily distinguish the log group from others. For example, **`CloudTrail/logs`**. Log group names can be between 1 and 512 characters long. Allowed characters include a-z, A-Z, 0-9, '_' (underscore), '-' (hyphen), '/' (forward slash), and '.' (period).

When CloudTrail creates the log stream for the log group, it names the log stream according to the following format: *`account_ID`*_CloudTrail_*`source_region`*.

Role Policy Document for CloudTrail to Use CloudWatch Logs for Monitoring

This section describes the trust policy required for the CloudTrail role to send log events to CloudWatch Logs. You can attach a policy document to a role when you configure CloudTrail to send events, as described in Sending CloudTrail Events to CloudWatch Logs (p. 50). You can also create a role using IAM. For more information, see Creating a Role for an AWS Service (AWS Management Console) or Creating a Role (CLI and API).

The following policy document contains the permissions required to create a CloudWatch log stream in the log group that you specify and to deliver CloudTrail events to that log stream. (This is the default policy for the default IAM role `CloudTrail_CloudWatchLogs_Role`.)

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

      "Sid": "AWSCloudTrailCreateLogStream2014110",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:accountID:log-group:log_group_name:log-
stream:CloudTrail_log_stream_name_prefix*"
      ]

    },
    {
      "Sid": "AWSCloudTrailPutLogEvents20141101",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:accountID:log-group:log_group_name:log-
stream:CloudTrail_log_stream_name_prefix*"
      ]
    }
  ]
}
```

Using the CloudTrail Processing Library

The CloudTrail Processing Library is a Java library that provides an easy way to process AWS CloudTrail logs in a fault-tolerant, scalable and flexible way. You provide configuration details about your CloudTrail SQS queue and write code to process events. The CloudTrail Processing Library does the rest, polling your Amazon SQS queue, reading and parsing queue messages, downloading CloudTrail log files, parsing events in the log files and passing them to your code as Java objects. The CloudTrail Processing Library is highly scalable and fault-tolerant, handling parallel processing of log files so that you can process as many logs as necessary, and robustly handling network failures related to network timeouts and inaccessable resources.

This chapter provides information about how to use the CloudTrail Processing Library to process CloudTrail logs in your Java projects. The library is provided as an Apache-licensed open-source project, available on GitHub:

- https://github.com/aws/aws-cloudtrail-processing-library

The library source includes sample code that you can use as a base for your own projects.

Topics

Minimum requirements

To use the CloudTrail Processing Library, you must have the following:

- **AWS SDK for Java 1.9.3**
- **Java 1.7**

Processing CloudTrail Logs with the CloudTrail Processing Library

To use the CloudTrail Processing Library to process CloudTrail logs in your Java application:

1. Add the CloudTrail Processing Library to your Project (p. 62)
2. Configure the CloudTrail Processing Library (p. 64)
3. Implement the Events Processor (p. 65)
4. Instantiate and Run the Processing Executor (p. 65)

Add the CloudTrail Processing Library to your Project

To use the CloudTrail Processing Library you must add it to your Java project's classpath.

Topics

Adding the Library to an Apache Ant Project

To add the CloudTrail Processing Library to an Ant project

1. Download or clone the CloudTrail Processing Library source code from GitHub at:

   - https://github.com/aws/aws-cloudtrail-processing-library

2. Build the .jar file from source as described in the README:

```
mvn clean install -Dgpg.skip=true
```

3. Copy the resulting .jar file into your project and add it to your project's `build.xml` file. For example:

```
<classpath>
  <pathelement path="${classpath}"/>
  <pathelement location="lib/aws-cloudtrail-processing-library-1.0.0.jar"/>
</classpath>
```

Adding the Library to an Apache Maven Project

The CloudTrail Processing Library is available for Apache Maven, so adding it to your project is as easy as writing a single dependency in your project's `pom.xml` file.

To add the CloudTrail Processing Library to a Maven project

- Using your favorite text editor, open your Maven project's `pom.xml` file and add the following dependency:

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-cloudtrail-processing-library</artifactId>
    <version>1.0.0</version>
</dependency>
```

Adding the CloudTrail Processing Library to an Eclipse Project

To add the CloudTrail Processing Library to an Eclipse project

1. Download or clone the CloudTrail Processing Library source code from GitHub at:
   - https://github.com/aws/aws-cloudtrail-processing-library
2. Build the .jar file from source as described in the README:

```
mvn clean install -Dgpg.skip=true
```

3. Copy the built aws-cloudtrail-processing-library-1.0.0.jar to a directory in your project (typically `lib`).
4. Right-click your project's name in the Eclipse **Project Explorer**, and select **Build Path** > **Configure**
5. In the **Java Build Path** window, click the **Libraries** tab.
6. Click **Add JARs...** and navigate to the path where you copied aws-cloudtrail-processing-library-1.0.0.jar.
7. Click **OK** to complete adding the `.jar` to your project.

Configure the CloudTrail Processing Library

You can configure the CloudTrail Processing Library by creating a classpath properties file that is loaded at runtime, or by creating a `ClientConfiguration` object and setting options manually.

Providing a Properties File

You can write a classpath properties file that provides configuration options to your application. Here is an example file that demonstrates the options you can set:

```
# AWS access key. (Required)
accessKey = your_access_key

# AWS secret key. (Required)
secretKey = your_secret_key

# The SQS URL used to pull CloudTrail notification from. (Required)
sqsUrl = your_sqs_queue_url

# The SQS end point specific to a region.
sqsRegion = us-east-1

# A period of time during which Amazon SQS prevents other consuming components
# from receiving and processing that message.
visibilityTimeout = 60

# The S3 region to use.
s3Region = us-east-1

# Number of threads used to download S3 files in parallel. Callbacks can be
# invoked from any thread.
threadCount = 1

# The time allowed, in seconds, for threads to shut down after
# AWSCloudTrailEventProcessingExecutor.stop() is called. If they are still
# running beyond this time, they will be forcibly terminated.
threadTerminationDelaySeconds = 60

# The maximum number of AWSCloudTrailClientEvents sent to a single invocation
# of processEvents().
maxEventsPerEmit = 10

# Whether to include raw event information in CloudTrailDeliveryInfo.
enableRawEventInfo = false
```

The required parameters are *`sqsUrl`*, *`accessKey`* and *`secretKey`*. The *`sqsUrl`* parameter provides the URL to pull your CloudTrail notifications from. If you don't provide this value, then an `IllegalStateException` will be thrown by the `AWSCloudTrailProcessingExecutor`. The *`accessKey`* and *`secretKey`* parameters provide your AWS credentials to the library, allowing it to access AWS on your behalf.

The other parameters have reasonable defaults that are set by the library. For information about the default values for each option, see the AWS CloudTrail Processing Library Reference.

Creating a ClientConfiguration

Instead of setting options in the classpath properties, you can provide options to the `AWSCloudTrailProcessingExecutor` by initializing and setting options on a `ClientConfiguration` object.

For example:

```
ClientConfiguration basicConfig = new ClientConfiguration(
    "http://sqs.us-east-1.amazonaws.com/123456789012/queue2",
    new DefaultAWSCredentialsProviderChain());

basicConfig.setEnableRawEventInfo(true);
basicConfig.setThreadCount(4);
basicConfig.setnEventsPerEmit(20);
```

Implement the Events Processor

To process CloudTrail logs, you must implement a `EventsProcessor` that receives the CloudTrail log data. Here is an example implementation:

```
public class SampleEventsProcessor implements EventsProcessor {

    public void process(List<CloudTrailEvent> events) {
        int i = 0;
        for (CloudTrailEvent event : events) {
            System.out.println(String.format("Process event %d : %s", i++,
event.getEventData()));
        }
    }
}
```

When implementing a `EventsProcessor`, you implement the `process()` callback that the `AWSCloudTrailProcessingExecutor` uses to send you CloudTrail events. Events are provided in a list of `CloudTrailClientEvent` objects.

The `CloudTrailClientEvent` object provides a `CloudTrailEvent` and `CloudTrailEventMetadata` that you can use to read the CloudTrail event and delivery info.

This simple example just prints the the event information for each event passed to `SampleEventsProcessor`. In your own implementation, you can process logs as you see fit. The `AWSCloudTrailProcessingExecutor` will continue to send events to your `EventsProcessor` as long as it has events to send and is still running.

Instantiate and Run the Processing Executor

Once you have written a `EventsProcessor` and have set configuration values for the CloudTrail Processing Library (either in a properties file or by using the `ClientConfiguration` class), you can use these elements to initialize and use a `AWSCloudTrailProcessingExecutor`.

To use `AWSCloudTrailProcessingExecutor` to process CloudTrail events

1. Instantiate an `AWSCloudTrailProcessingExecutor.Builder` object. `Builder`'s constructor takes a `EventsProcessor` object and a classpath properties file name.
2. Call the `Builder`'s `build()` factory method to configure and obtain an AWSCloudTrailProcessingExecutor object.
3. Use the `AWSCloudTrailProcessingExecutor`'s `start()` and `stop()` methods to begin and end CloudTrail event processing.

```
public class SampleApp {
  public static void main(String[] args) throws InterruptedException {
    AWSCloudTrailProcessingExecutor executor = new
      AWSCloudTrailProcessingExecutor.Builder(new SampleEventsProcessor(),
        "/myproject/cloudtrailprocessing.properties").build();

    executor.start();
    Thread.sleep(24 * 60 * 60 * 1000); // let it run for a while (optional)
    executor.stop(); // optional
  }
}
```

Advanced Topics

Topics

Filtering the Events to Process

By default, all of the logs in your Amazon SQS queue's S3 bucket and all of the events that they contain will be sent to your `EventsProcessor`. The CloudTrail Processing Library provides optional interfaces that you can implement to filter the sources used to obtain CloudTrail logs and to filter the events that you are interested in processing.

SourceFilter

You can implement the `SourceFilter` interface to choose whether or not you want to process logs from a provided source. `SourceFilter` declares a single callback method, `filterSource()`, that receives a `CloudTrailSource` object. To keep events from a source from being processed, return `false` from `filterSource()`.

The `filterSource()` method is called by the CloudTrail Processing Library after the library has polled for logs on the Amazon SQS queue, but before event filtering or processing has been done for those logs.

Here is an example implementation:

```
public class SampleSourceFilter implements SourceFilter{
  private static final int MAX_RECEIVED_COUNT = 3;

  private static List<String> accountIDs ;
  static {
    accountIDs = new ArrayList<>();
    accountIDs.add("123456789012");
    accountIDs.add("234567890123");
  }

  @Override
  public boolean filterSource(CloudTrailSource source) throws CallbackExcep
tion {
    source = (SQSBasedSource) source;
    Map<String, String> sourceAttributes = source.getSourceAttributes();
```

```
    String accountId = sourceAttributes.get(
      SourceAttributeKeys.ACCOUNT_ID.getAttributeKey());

    String receivedCount = sourceAttributes.get(
      SourceAttributeKeys.APPROXIMATE_RECEIVE_COUNT.getAttributeKey());

    int approximateReceivedCount = Integer.parseInt(receivedCount);

   return approximateReceivedCount <= MAX_RECEIVED_COUNT && accountIDs.con
tains(accountId);
  }
}
```

If you don't provide your own `SourceFilter`, then `DefaultSourceFilter` will be used, which allows all sources to be processed (it always returns `true`).

EventFilter

You can implement the `EventFilter` interface to choose whether a CloudTrail event will be sent to your `EventsProcessor` or not. `EventFilter` declares a single callback method, `filterEvent()`, that receives a `CloudTrailEvent` object. To keep the event from being processed, return `false` from `filterEvent()`.

The `filterEvent()` method is called by the CloudTrail Processing Library after the library has polled for logs on the Amazon SQS queue and after source filtering, but before event processing has been done for those logs.

Here is an example implementation:

```
public class SampleEventFilter implements EventFilter{

  private static final String EC2_EVENTS = "ec2.amazonaws.com";

  @Override
  public boolean filterEvent(CloudTrailClientEvent clientEvent) throws
CallbackException {
    CloudTrailEvent event = clientEvent.getEvent();

    String eventSource = event.getEventSource();
    String eventName = event.getEventName();

   return eventSource.equals(EC2_EVENTS) && eventName.startsWith("Delete");

  }
}
```

If you don't provide your own `EventFilter`, then `DefaultEventFilter` will be used, which allows all events to be processed (it always returns `true`.

Reporting Progress

The `ProgressReporter` interface can be implemented to customize the reporting of CloudTrail Processing Library progress. `ProgressReporter` declares two methods: `reportStart()` and `reportEnd()`, which are called at the beginning and end of the following operations:

- polling messages from Amazon SQS.
- parsing messages from Amazon SQS.

- processing an Amazon SQS source for CloudTrail logs.
- deleting messages from Amazon SQS.
- downloading a CloudTrail log file.
- processing a CloudTrail log file.

Both methods receive a `ProgressStatus` object that contains information about the operation being performed (in the `progressState` member, which holds a member of the `ProgressState` enumeration that identifies the current operation) and can contain additional info (in the `progressInfo` member). Additionally, any object that you return from `reportStart()` will be passed to `reportEnd()`, so you can provide contextual information such as what time it was when the event began processing.

Here is an example implementation that provides information about how long an operation took to complete:

```
public class SampleProgressReporter implements ProgressReporter {
  private static final Log logger =
    LogFactory.getLog(DefaultProgressReporter.class);

  @Override
  public Object reportStart(ProgressStatus status) {
    return new Date();
  }

  @Override
  public void reportEnd(ProgressStatus status, Object startDate) {
    System.out.println(status.getProgressState().toString() + " is " +
      status.getProgressInfo().isSuccess() + " , and latency is " +
      Math.abs(((Date) startDate).getTime()-new Date().getTime()) + "
      milliseconds.");
  }
}
```

If you don't implement your own `ProgressReporter`, then `DefaultExceptionHandler`, which prints the name of the state being run, will be used instead.

Handling Errors

The `ExceptionHandler` interface allows you to provide special handling when an exception occurs during log processing. `ExceptionHandler` declares a single callback method, `handleException()`, which receives a `ProcessingLibraryException` object with context about the exception that occured.

You can use the passed-in `ProcessingLibraryException`'s `getStatus()` method to find out what operation was being executed when the exception occured and get additional info about the status of the operation. `ProcessingLibraryException` is derived from Java's standard `Exception` class, so you can also retrieve information about the exception by invoking any of the Exception methods, as well.

Here is an example implementation:

```
public class SampleExceptionHandler implements ExceptionHandler{
  private static final Log logger =
    LogFactory.getLog(DefaultProgressReporter.class);

  @Override
  public void handleException(ProcessingLibraryException exception) {
    ProgressStatus status = exception.getStatus();
    ProgressState state = status.getProgressState();
```

```
    ProgressInfo info = status.getProgressInfo();

    System.err.println(String.format(
     "Exception. Progress State: %s. Progress Information: %s.", state, info));

  }
}
```

If you don't provide your own `ExceptionHandler`, then `DefaultExceptionHandler`, which simply prints a standard error message, will be used instead.

Additional Resources

For more information about the CloudTrail Processing Library, see the following additional resources:

- The CloudTrail Processing Library GitHub project, which includes sample code that demonstrates how to implement a CloudTrail Processing Library application
- The CloudTrail Processing Library Java Package Documentation

CloudTrail Event Reference

A CloudTrail log is a record in JSON format that contains information about requests for resources in your account—what service was accessed, what action was performed, and any parameters for the action. The request also helps you determine who made the request. The event data is enclosed in a `Records` array.

The following example shows a single log record at the beginning of a log file. The entry indicates that an IAM user named Alice called the CloudTrail **StartLogging** API by using the CloudTrail console to start the logging process.

```
{
 "Records": [{
  "eventVersion": "1.01",
  "userIdentity": {
   "type": "IAMUser",
   "principalId": "AIDAJDPLRKLG7UEXAMPLE",
   "arn": "arn:aws:iam::123456789012:user/Alice",
   "accountId": "123456789012",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "userName": "Alice",
   "sessionContext": {
    "attributes": {
     "mfaAuthenticated": "false",
     "creationDate": "2014-03-18T14:29:23Z"
    }
   }
  },
  "eventTime": "2014-03-18T14:30:07Z",
  "eventSource": "cloudtrail.amazonaws.com",
  "eventName": "StartLogging",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "72.21.198.64",
  "userAgent": "AWSConsole, aws-sdk-java/1.4.5 Linux/x.xx.fleetxen Java_Hot
Spot(TM)_64-Bit_Server_VM/xx",
  "requestParameters": {
   "name": "Default"
  },
  "responseElements": null,
```

```
 "requestID": "cdc73f9d-aea9-11e3-9d5a-835b769c0d9c",
 "eventID": "3074414d-c626-42aa-984b-68ff152d6ab7"
},
   ... additional entries ...
]
```

The following topics list the fields that CloudTrail uses to represent the data it captures for each AWS API call and sign-in event.

Topics

Record Body Contents

The body of the record contains fields that you can use to determine what action was requested and when and where the request was made.

eventTime
The date and time the request was made, in coordinated universal time (UTC).

eventVersion
The version of the log event format. The current version is 1.02.

userIdentity
Information about the user that made a request. For more information, see userIdentity Element (p. 73)

eventSource
The service that the request was made to. For example, a call to Amazon EC2 is listed in the `eventSource` field as `ec2.amazonaws.com`.

eventName
The requested action, which is one of the actions listed in the API Reference for the service.

awsRegion
The AWS region that the request was made to.

sourceIPAddress
The apparent IP address that the request was made from. For actions that originate from the service console, the address reported is for the underlying customer resource, not the console web server. For services in AWS, only the DNS name is displayed.

userAgent
The agent through which the request was made, such as the AWS Management Console or an AWS SDK. For example:

- `aws-cli/1.1.1 Python/2.7.4 Windows/7`. The request was made using the AWS CLI installed on Microsoft Windows 7.
- `AWSConsole`. The request was made through the AWS Management Console.

errorCode
The AWS service error if the request returns an error.

errorMessage
If the request returns an error, the description of the error. This message includes messages for authorization failures. For such messages, CloudTrail captures the message logged by the service in its exception handling.

> **Note**
> Some AWS services provide the errorCode and errorMessage as top-level fields in the event. Other AWS services include error information as part of responseElements.

requestParameters
The parameters, if any, that were sent with the request. These are fully documented in the API Reference documentation for the appropriate AWS service.

responseElements
The response element for actions that make changes (create, update, or delete actions). If an action does not change state (for example, a request to get or list objects), this element is omitted. These are fully documented in the API Reference documentation for the appropriate AWS service.

requestID
GUID generated by the service being called to uniquely identify the request.

Support for this field begins with **eventVersion** `1.01`.

eventID
GUID generated by CloudTrail to uniquely identify each event. You can use this value to identify a single event. For example, you can use the ID as a primary key to retrieve log data from a searchable database.

Support for this field begins with **eventVersion** `1.01`.

eventType
Identifies the type of event that generated the event record. Currently, this can be the following value.

- `AwsApiCall` - An API was called.

Support for this field begins with **eventVersion** `1.02`.

apiVersion
Identifies the API version number associated with the `AwsApiCall` **eventType** value.

Support for this field begins with **eventVersion** `1.02`.

recipientAccountID
Represents the account ID that received this event. Currently this will be the same as the userIdentity Element (p. 73) **accountId** value.

Support for this field begins with **eventVersion** `1.02`.

userIdentity Element

Information about the user that made a request is included in the `userIdentity` element. This information can help you determine how your AWS resources have been accessed. The `userIdentity` element can tell you details such as the following:

- What type of principal (root user, IAM user, federated user, AWS service, etc.) made the request.
- Whether the request was made with root or IAM user credentials, or with temporary security credentials for a role, federated user, or by another service in AWS.
- For temporary security credentials, how the credentials were obtained.
- Whether the request was made using the AWS Management Console.
- Whether the request was made by another AWS service.
- Whether the request was made using web identity federation.

The following example shows the `userIdentity` element of a simple request made with the credentials of the IAM user named `Alice`.

```
"userIdentity": {
  "type": "IAMUser",
  "principalId": "AIDAJ45Q7YFFAREXAMPLE",
  "arn": "arn:aws:iam::123456789012:user/Alice",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Alice"
}
```

The following example shows a `userIdentity` element for a request made using temporary security credentials obtained by assuming an IAM role. In this case, the element contains additional details about the role that was assumed to get credentials.

```
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
    "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/MySession
Name",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "creationDate": "  20131102T010628Z  ",
        "mfaAuthenticated": "false"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIDPPEZS35WEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/RoleToBeAssumed",
        "accountId": "123456789012",
        "userName": "RoleToBeAssumed"
      }
    }
}
```

The following fields can appear in a `userIdentity` element.

type
The type of the principal that made the call:

- `Root`. The request was made using root account credentials.
- `IAMUser`. The request was made using the credentials of an IAM user.
- `AssumedRole`. The request was made with temporary security credentials that were obtained via a role; specifically, via a call to the AWS STS `AssumeRole` API. This can include roles for Amazon EC2 and cross-account API access.

  The `type` field is also set to `AssumedRole` when a request was made by a mobile app that authenticated the user with web identity federation, using the `AssumeRoleWithWebIdentity` API.
- `FederatedUser`. The request was made with temporary security credentials that were obtained via a call to the AWS STS `GetFederationToken` API. The `sessionIssuer` element indicates whether the API was called using root or IAM user credentials.

principalId
A unique identifier for the principal. For requests made with temporary security credentials, this value includes the session name that is passed to the `AssumeRole`, `AssumeRoleWIthWebIdentity`, or `GetFederationToken` API call.

arn
The Amazon Resource Name (ARN) of the principal that made the call.

accountId
The account that owns the entity that granted permissions for the request. If the request was made using temporary security credentials, this is the account that owns the IAM user or role that was used to obtain credentials.

accessKeyId
The access key ID that was used to sign the request. If the request was made using temporary security credentials, this is the access key ID of the temporary credentials.

userName
Friendly name of the principal that made the call.

sessionContext
If the request was made with temporary security credentials, an element that provides information about the session that was created for those credentials. Sessions are created when any API is called that returns temporary credentials. Sessions are also created when users work in the console and when users make a request using APIs that include multi-factor authentication. Attributes for this element are:

- **creationDate**. The date and time when the temporary security credentials were issued. Represented in ISO 8601 basic notation.
- **mfaAuthenticated**. `true` if the root user or IAM user whose credentials were used for the request also was authenticated using an MFA device; otherwise, `false`.

invokedBy
If the request was made by another AWS service, such as Auto Scaling or AWS Elastic Beanstalk, the name of the service.

sessionIssuer
If the request was made with temporary security credentials, an element that provides information about how the credentials were obtained. For example, if the temporary security credentials were obtained by assuming a role, this element provides information about the assumed role. If the credentials were obtained by using root or IAM user credentials to call `GetFederationToken`, the element provides information about the root account or IAM user. Attributes for this element are:

- **type**. The source of the temporary security credentials, such as "Root", "IAMUser", or "Role".
- **principalId**. The internal ID of the entity that was used to get credentials.
- **arn** The ARN of the source (account, IAM user, or role) that was used to get temporary security credentials.
- **accountId**. The account that owns the entity that was used to get credentials.
- **userName**. The friendly name of the user or role.

webIdFederationData

If the request was made with temporary security credentials obtained using web identity federation, an element that lists information about the identity provider. Attributes for this element are:

- **federatedProvider**. The principal name of the identity provider (for example, `www.amazon.com` for Login with Amazon or `accounts.google.com` for Google).
- **attributes**. The application ID and user ID as reported by the provider (for example, `www.amazon.com:app_id` and www.amazon.com:user_id for Login with Amazon. For more information, see Available Keys for Web Identity Federation in the *Using IAM* guide.

Non-API Events

In addition to logging AWS API calls, CloudTrail also captures other related events that might have a security or compliance impact on your AWS account or that might help you troubleshoot operational problems.

Topics

AWS Console Sign-in Events

CloudTrail records attempts to sign into the AWS Management Console, the AWS Discussion Forums and the AWS Support Center. All IAM user sign-in attempts (successes and failures), all federated user sign-in events (successes and failures) and all successful AWS root account sign-in attempts generate records in CloudTrail log files. Note, however, that CloudTrail does not record root sign-in failures.

The following record shows that an IAM user named Alice successfully signed into the AWS console without using multi-factor authentication.

```
{
   "Records":[
      {
         "eventVersion":"1.01",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDAELOPP77CWZEXAMPLE",
            "arn":"arn:aws:iam::12345679012:user/alice",
            "accountId":"12345679012",
            "userName":"alice"
         },
         "eventTime":"2014-07-08T17:35:32Z",
         "eventSource":"signin.amazonaws.com",
         "eventName":"ConsoleLogin",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4b)
 Gecko/20030516 Mozilla Firebird/0.6",
         "requestParameters":null,
         "responseElements":{
            "ConsoleLogin":"Success"
         },
         "additionalEventData":{
            "MobileVersion":"No",
```

```
            "LoginTo":"https://console.aws.amazon.com/sns",
            "MFAUsed":"No"
         },
         "eventID":"3fcfb182-98f8-4744-bd45-10a395ab61cb"
      }
   ]
}
```

The following record shows that an IAM user named Alice logged into the AWS console by using multi-factor authentication.

```
{
   "Records":[
      {
         "eventVersion":"1.01",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDAEZ7VBM6PDZEXAMPLE",
            "arn":"arn:aws:iam::12345679012:user/Alice",
            "accountId":"12345679012",
            "userName":"Alice"
         },
         "eventTime":"2014-07-08T17:36:03Z",
         "eventSource":"signin.amazonaws.com",
         "eventName":"ConsoleLogin",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4b)
 Gecko/20030516 Mozilla Firebird/0.6",
         "requestParameters":null,
         "responseElements":{
            "ConsoleLogin":"Success"
         },
         "additionalEventData":{
            "MobileVersion":"Yes",
            "LoginTo":"https://console.aws.amazon.com/sns",
            "MFAUsed":"Yes"
         },
         "eventID":"5d2c2f55-3d1e-4336-b940-dbf8e66f588c"
      }
   ]
}
```

The following record shows an unsuccessful AWS console sign-in attempt because of an authentication failure.

```
{
   "Records":[
      {
         "eventVersion":"1.01",
         "userIdentity":{
            "type":"IAMUser",
            "principalId":"AIDAELOPP77CWZEXAMPLE",
```

```
            "accountId":"12345679012",
            "accessKeyId":"",
            "userName":"alice"
         },
         "eventTime":"2014-07-08T17:35:27Z",
         "eventSource":"signin.amazonaws.com",
         "eventName":"ConsoleLogin",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"192.0.2.0",
         "userAgent":"Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4b)
 Gecko/20030516 Mozilla Firebird/0.6",
         "errorMessage":"Failed authentication",
         "requestParameters":null,
         "responseElements":{
            "ConsoleLogin":"Failure"
         },
         "additionalEventData":{
            "MobileVersion":"No",
            "LoginTo":"https://console.aws.amazon.com/sns",
            "MFAUsed":"No"
         },
         "eventID":"11ea990b-4678-4bcd-8fbe-62509088b7cf"
      }
   ]
}
```

Document History

The following table describes the documentation release history of AWS CloudTrail.

- **API version**: 2013-11-01
- **Latest documentation update**: November 10, 2014

Change	Description	Release Date
New Guide	This release introduces AWS CloudTrail.	November 13, 2013
New content	The release includes information about aggregating log files to a single Amazon S3 bucket. See Aggregating Log Files from Multiple Regions (p. 33).	December 23, 2013
Additional service support	This release supports AWS CloudFormation. See Deployment and Management (p. 7).	March 7, 2014
Added service support	This release supports AWS Elastic Beanstalk. See Deployment and Management (p. 7).	April 2, 2014
Added service support	This release supports Amazon Elastic MapReduce. See Analytics (p. 4).	April 4, 2014
Added service support	This release supports AWS Direct Connect. See Compute and Networking (p. 6).	April 11, 2014
Added service support	This release supports Amazon Kinesis. See Analytics (p. 4).	April 22, 2014
Added service support	This release supports Amazon CloudWatch. See Deployment and Management (p. 7).	April 28, 2014
Added new content	This release includes topics that discuss sharing log files between accounts. See Sharing CloudTrail Log Files Between AWS Accounts (p. 40).	May 2, 2014
Added service support	This release supports Amazon Simple Workflow Service. See App Services (p. 5).	May 9, 2014

Change	Description	Release Date
Added region support	This release supports three additional regions: us-west-1 (Northern California), eu-west-1 (Ireland), ap-southeast-2 (Sydney). See Supported Regions (p. 9).	May 13, 2014
Added service support	This release supports Amazon CloudFront. See Storage and Content Delivery (p. 8).	May 28, 2014
Added service support	This release supports AWS OpsWorks. See Deployment and Management (p. 7).	June 5, 2014
Added region support	This release supports three additional regions: ap-southeast-1 (Singapore), ap-northeast-1 (Tokyo), sa-east-1 (Sao Paulo). See Supported Regions (p. 9).	June 30, 2014
Added service support	This release supports Auto Scaling (see Compute and Networking (p. 6)) and Amazon SQS (see App Services (p. 5)).	July 17, 2014
Added new content	The **eventVersion** element for this release has been upgraded to version 1.02 and three new fields have been added. See Record Body Contents (p. 71).	July 18, 2014
Added new content	This release includes a topic that discusses logging sign-in events. See AWS Console Sign-in Events (p. 75).	July 24, 2014
Added service support	This release supports Amazon Zocalo. See Applications (p. 5).	August 26, 2014
Added service support	This release supports Amazon ElastiCache. See Database (p. 6).	September 15, 2014
Added service support	This release supports Amazon Simple Notification Service. See Mobile Services (p. 8).	October 09, 2014
Added service support	This release supports Amazon CloudSearch. See App Services (p. 5).	October 16, 2014
Added region support	This release supports one additional region: eu-central-1 (Frankfurt). See Supported Regions (p. 9).	October 23, 2014
Added service support	This release supports Amazon Elastic Transcoder. See App Services (p. 5).	October 27, 2014
New documentation	A new section, Using the CloudTrail Processing Library (p. 61), has been added to the guide. It provides information about how to write a CloudTrail log processor in Java using the AWS CloudTrail Processing Library.	November 5, 2014
New documentation	A new section, Monitoring CloudTrail Log Files with Amazon CloudWatch Logs (p. 50), has been added to the guide. It describes how to use Amazon CloudWatch Logs to monitor CloudTrail log events.	November 10, 2014
Added service support	This release supports AWS Key Management Service. See Deployment and Management (p. 7).	November 12, 2014