Amazon Fraud Detector

User Guide

Version latest

Amazon Fraud Detector: User Guide

Table of Contents

What Is Amazon Fraud Detector?

Amazon Fraud Detector is a fully managed service that makes it easy to identify potentially fraudulent online activities such as online payment fraud and the creation of fake accounts. Amazon Fraud Detector uses your data, machine learning (ML), and more than 20 years of fraud detection expertise from Amazon to automatically identify potentially fraudulent online activity so you can catch more fraud faster.

Amazon Fraud Detector enables you to build fraud-detection models. You don't need ML experience to start using Amazon Fraud Detector, you only need to provide your company's historical fraud data. Amazon Fraud Detector uses this data to automatically train, test, and deploy a customized fraud detection model. During this process, a series of models that have learned patterns of fraud from AWS and Amazon's own fraud expertise are used to boost your model's performance. To use the model, call the Amazon Fraud Detector `GetEventPrediction` API with meta data about an online event and synchronously receive a fraud prediction score and an outcome based on your configuration.

Amazon Fraud Detector Overview

You don't need ML experience to use Amazon Fraud Detector. Simply upload your data to train, test, and deploy a fraud-detection model. Deployed models are imported to detectors, where you can configure decision logic (for example rules) to interpret the model's score and assign outcomes such as pass or send transaction to a human investigator for review.

To use the model, call the Amazon Fraud Detector `GetEventPrediction` API with meta data about an online event. Amazon Fraud Detector synchronously returns a fraud prediction score and an outcome based on your detector configuration. For example, you can send Amazon Fraud Detector metadata for an online transaction event, such as price, product name, and shipping address of the purchasing entity, and determine the appropriate action for the transaction based on your deployed model and detector configuration

> **Note**
> Amazon Fraud Detector is available in the US East (N. Virginia), US East (Ohio), US West (Oregon), Europe (Ireland), Asia Pacific (Singapore) and Asia Pacific (Sydney) AWS regions.

How to Use this Guide

If you are a new user of Amazon Fraud Detector, read the following sections to get started:

1. How Amazon Fraud Detector works (p. 2) – Introduction to Amazon Fraud Detector concepts that you use to create fraud detection models and fraud evaluations.
2. Setting up Amazon Fraud Detector (p. 4) – Prerequisite steps to get started with Amazon Fraud Detector.
3. Get started (console) (p. 6) – Tutorial that walks you through how to use Amazon Fraud Detector.

For information about Amazon Fraud Detector the API operations, see the Amazon Fraud Detector API Reference.

How Amazon Fraud Detector works

To generate fraud predictions, Amazon Fraud Detector uses machine learning models that are trained with your historical fraud data. Each model is trained using a model type, which is a specialized recipe to build a fraud detection model for a specific fraud use case. Deployed models are imported to detectors, where you can configure decision logic (for example, rules) to interpret the model's score and assign outcomes such as pass or send transaction to a human investigator for review.

You can use the AWS Console to create and manage models and detector versions. Alternatively, you can use the AWS Command Line Interface (AWS CLI) or one of the Amazon Fraud Detector SDKs.

Amazon Fraud Detector components include events, entities, labels, models, rules, variables, outcomes, and detectors. Using these components, you can build an evaluation that contains your fraud detection logic.

For additional information about Amazon Fraud Detector concepts and how they operate, see Amazon Fraud Detector concepts (p. 2).

Amazon Fraud Detector workflow

The steps for creating a model, building a detector, and getting fraud predictions include:

1. Define the event you want to evaluate for fraud.
2. Gather historical event data (training data).
3. Create a model version (trained model) using a model type.
4. Create a detector that includes the model version and decision rules.
5. Send events to Amazon Fraud Detector and get a fraud prediction.

Amazon Fraud Detector concepts

Key concepts for understanding Amazon Fraud Detector include:

Events and Event Types

An event is a business activity that is evaluated for fraud risk. With Amazon Fraud Detector, you generate fraud predictions for events.

An event type defines the structure for an event sent to Amazon Fraud Detector. This includes the variables sent as part of the event, the entity performing the event (such as a customer), and the labels that classify the event. Example event types include online payment transactions, account registrations, and authentication.

Entity and Entity Type

An entity represents who is performing the event. As part of a fraud prediction, you can pass the entity ID to indicate the specific entity who performed the event.

An entity type classifies the entity. Example classifications include customer, merchant, or account.

Label

A label classifies an event as fraudulent or legitimate. Labels are used to train supervised machine learning models in Amazon Fraud Detector.

Model type

The model type defines the algorithms, enrichments, and feature transformations used during model training as well as the data requirements to train the model.

Model training and Model Versions

Model training is the process of using a provided dataset to create a model that can predict fraudulent events. All steps in the model training process are fully automated including data validation, data transformation, feature engineering, algorithm selection, training, and model optimization. In Amazon Fraud Detector, the output from training is called a *model version*.

Model deployment

When you deploy a model, you make a trained model version available for real-time fraud predictions. Only deployed model versions can be imported to detectors.

Amazon SageMaker model endpoint

In addition to building models using Amazon Fraud Detector, you can optionally use SageMaker-hosted model endpoints in Amazon Fraud Detector evaluations.

For more information about building a model in SageMaker, see Train a Model with Amazon SageMaker.

Detector

A detector contains the detection logic (such as the models and rules) for a particular event that you want to evaluate for fraud. A detector can have multiple versions, with each version having a status of `Draft`, `Active`, or `Inactive`. Only one detector version can be in Active status at a time.

Variable

A variable represents a data element associated with an event that you want to use in a fraud prediction. Variables can either be sent with an event as part of a fraud prediction or derived, such as the output of an Amazon Fraud Detector model or Amazon SageMaker model.

Rule

A rule is a condition that tells Amazon Fraud Detector how to interpret variable values during a fraud prediction. A rule consists of one or more variables, a logic expression, and one or more outcomes. The variables used in the rule must be part of the event evaluated by the detector. A detector must have at least one associated rule.

Outcome

This is the result, or output, from a fraud prediction. Each rule used in a fraud prediction must specify one or more outcomes.

Prediction

After you create a detector, you can generate fraud predictions by calling the `GetEventPrediction` API with your event variables and specifying which detector you want to use. The API always returns an outcome. If a model is used in the detector, Amazon Fraud Detector also returns the corresponding fraud prediction score.

Setting up Amazon Fraud Detector

Before using Amazon Fraud Detector, you must have an Amazon Web Service (AWS) account. After you have an AWS account, you can access Amazon Fraud Detector console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

After you complete these steps, see Get started (console) (p. 6) to continue getting started with Amazon Fraud Detector.

Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon Fraud Detector. You are charged only for the services that you use. If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open https://aws.amazon.com and then choose **Create an AWS Account**.
2. Follow the on-screen instructions to complete the account creation. Note your **12-digit AWS account number**.

Setting up permissions

To use Amazon Fraud Detector, you have to set up permissions that allow access to the Amazon Fraud Detector console and API operations. You also have to allow Amazon Fraud Detector to perform tasks on your behalf and to access resources that you own.

We recommend creating an AWS Identify and Access Management (IAM) user with access restricted to Amazon Fraud Detector operations and required permissions. You can add other permissions as needed.

The following policies provide the required permission to use Amazon Fraud Detector:

- `AmazonFraudDetectorFullAccessPolicy`

  Allows you to perform the following actions:
  - Access all Amazon Fraud Detector resources
  - List and describe all model endpoints in SageMaker
  - List all IAM roles in the account
  - List all Amazon S3 buckets
  - Allow IAM Pass Role to pass a role to Amazon Fraud Detector
- `AmazonS3FullAccess`

  Allows full access to Amazon S3. This is required if you need to upload training datasets to S3.

Create an IAM user and assign required permissions

The following describes how to create an IAM user and assign the needed permissions.

To create an IAM user and assign required permissions

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **`AmazonFraudDetectorUser`**.
4. Select the **AWS Management Console access** check box, and then configure the user's password.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Choose **Create group**.
8. For **Group name** enter **`AmazonFraudDetector Group`**.
9. In the policy list, select the check box for **AmazonFraudDetectorFullAccessPolicy** and **AmazonS3FullAccess**. Choose **Create group**.
10. In the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
11. Choose **Next: Tags**.
12. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM Users and Roles.
13. Choose **Next: Review** to see the **User details** and **Permissions summary** for the new user. When you are ready to proceed, choose **Create user**.

Get started (console)

Before you start this exercise, you must complete the steps in Setting up Amazon Fraud Detector (p. 4).

This exercise consists of the following parts:

- In Part A: Build an Amazon Fraud Detector model (p. 6), you create a registration event and train a model by associating training data with the Amazon Fraud Detector Online Fraud Insights model type. Then, you train and deploy the model.
- In Part B: Generate real-time fraud predictions (p. 8), you create a detector to evaluate registrations events. You will add your deployed model, write rules to interpret the model output, and generate a fraud prediction.

Part A: Build an Amazon Fraud Detector model

The procedures in this section provide the steps for you to successfully train and deploy a fraud-detection model.

Topics

Get and upload example training data

1. Download the following file, unzip and use one of the sample CSV files that contain fictitious, synthetically generated training data.

   Download: Training_Data.zip.

   This zip file contains two files of synthetic registrations that you can use to train a model. The dataset registration_data_20K_minimum contains only two variables: ip_address and email_address. The dataset registration_data_20K_full contains additional variables for each event such as billing_address, phone_number, and user_agent. Both datasets also contains two mandatory fields:

   - EVENT_TIMESTAMP – Defines when the event occurred
   - EVENT_LABEL – Classifies the event as fraudulent or legitimate
2. Create an Amazon S3 bucket:

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. Choose **Create bucket**, and perform the steps to create your bucket. You must choose an AWS region where Amazon Fraud Detector is currently available: US East (N. Virginia), US East (Ohio), US West (Oregon), Europe (Ireland), Asia Pacific (Singapore) or Asia Pacific (Sydney).

      For this exercise, the generic name *bucket-name* is used. You must rename your bucket because Amazon S3 bucket names must be globally unique.

3. Upload a training data file (that is, one of the .csv files listed previously) to your Amazon S3 bucket.

   Note the Amazon S3 location of your training file (for example, *s3://bucketname/path/to/some/object.csv*) and your role name. For details about formatting your dataset file, see Preparing training data (p. 17).

Step 1: Define event to evaluate for fraud

1. Open the AWS Console and sign in to your account. Navigate to Amazon Fraud Detector.
2. In the left navigation pane, choose **Events**.
3. On the **Events** page, choose **Create**.
4. Enter `sample_registration` as the event type name.
5. In the Entity drop-down, select **Create entity**.
6. In **Create entity type**, enter `sample_customer` as the entity type name and, optionally, enter a description of the entity type.
7. Choose **Create entity**.
8. For **Event variables**, choose **Select variables from a training dataset**.
9. In **IAM role**, select **Create IAM role**. On the **Create IAM role** page, enter the specific bucket name where you uploaded your training data. Choose **Create role**.
10. In **Data location**, enter the path to your training data. Then choose **upload**. Amazon Fraud Detector will extract the headers from your training dataset.
11. To allow Amazon Fraud Detector to interpret your variables, you must map them to variable types. Map `ip_address` to "IP Address" and `email_address`, to "Email Address".
12. For **Labels**, choose **Create new labels**.
13. In **Create label**, enter `fraud` as the label name as this label corresponds to the value that represents fraudulent events in the synthetic dataset.
14. Choose **Create label**.
15. Create a second label by following the same steps, this time entering `legit` as the label name as this corresponds to the value that represents legitimate events in the synthetic dataset.
16. Choose **Create event type**.
17. In **Build a model**, choose **Build a model**.

Step 2: Define model details

1. On the **Models** page, choose **Add model** and then choose **Create model**.
2. On **Step 1 – Define model details**, enter `sample_fraud_detection_model` as the model name and, optionally, enter a description of the model.
3. For **Model Type**, choose the **Online Fraud Insights** model.
4. For **Event type**, choose **sample_registration** (the event type you created in Step 1).
5. In **Historical event data**, for **IAM role**, select the role you created in Step 1.
6. In **Training data location**, enter the path to your training data.
7. Choose **Next**.

Step 3: Configure training and train model

1. In **Model inputs**, leave all checkboxes checked. By default, Amazon Fraud Detector will use all variables from your historical event dataset as model inputs.

2. In **Label classification**, for **Fraud labels** choose **fraud** as this label corresponds to the value that represents fraudulent events in the synthetic dataset. For **Legitimate labels**, choose **legit** as this label corresponds to the value that represents legitimate events in the synthetic dataset.
3. Choose **Next**.
4. After reviewing, choose **Create and train model**. Amazon Fraud Detector will:

   - Create the model output variable that can be used to write rules
   - Create the model
   - Begin to train a new version of the model

   Model training using the example training data set takes approximately 45 minutes to complete.

Step 4: Review the trained model's performance

An important step in using Amazon Fraud Detector is to assess the accuracy of your model using model scores and performance metrics.

After model training is complete, Amazon Fraud Detector validates model performance using 15% of your data that was not used to train the model. You can expect your trained Amazon Fraud Detector model to have real-world fraud detection performance that is similar to the validation performance metrics.

To learn more about model scores and model performance metrics, see Model scores (p. 21) and Training performance metrics (p. 22).

Step 5: Deploy the model

After you've validated your trained model and are ready to use it in real-time fraud predictions, you can deploy the model.

1. In the Amazon Fraud Detector console's left navigation pane, choose **Models**.
2. In the **Models** page, choose **sample_fraud_detection_model**, and then choose the **specific model version** that you want to deploy (for example, version 1.0).
3. On the **Model version** page, choose **Actions** and then choose **Deploy model version**. The model version is now available to add to detectors, which is covered in Part B: Generate real-time fraud predictions (p. 8).

Part B: Generate real-time fraud predictions

In Amazon Fraud Detector, you create and configure detectors to hold your deployed model and decision logic (that is, rules). In Part B, You create rules for your detector. These rules outcomes (such as to flag an e-commerce transaction if the ML score is too high).

To get a fraud prediction for an event, you send metadata about an online event using the `GetEventPrediction` API and then synchronously receive a fraud prediction outcome and model score. In Amazon Fraud Detector, you configure fraud prediction logic using these components: event types, detectors, models, rules, variables, and outcomes.

To create your fraud prediction logic in Amazon Fraud Detector, follow the steps in this section.

Topics

Step 1: Create a detector

You use a detector to house your fraud prediction configurations.

1. In the Amazon Fraud Detector console's left navigation pane, choose **Detectors**.
2. Choose **Create detector**.
3. On **Step 1 – Define detector details**, enter `sample_detector`, and optionally enter a description for the detector, such as `my sample fraud detector`.
4. For **Event Type**, select **sample_registration** (the event you created in Part A of this exercise).
5. Choose **Next**.

Step 2: (Optional) Add a model to a detector

If you completed Part A of the Get Started exercise, you should already have an Amazon Fraud Detector model available to add to your detector. If you haven't created a model yet, go to the model library, configure and train a new model, and then deploy it.

1. On **Step 2 – Add model**, choose **Add Model**, choose the Amazon Fraud Detector model name and version you created and deployed in Part A of the Get Started exercise from the list, and then choose **Add model**.
2. Choose **Next**.

Step 3: Add rules to a detector

After you have named your detector and added a model, you can create rules to interpret your Amazon Fraud Detector model's score. For this exercise, you create three rules: high_fraud_risk, medium_fraud_risk, and low_fraud_risk.

1. On **Step 3 – Add rules**, enter `high_fraud_risk` for the rule name under **Define a rule**, and enter **`This rule captures events with a high ML model score`** as the description for the rule.
2. In **Expression**, enter the following rule expression using the Amazon Fraud Detector simplified rule expression language:

   ```
   $sample_fraud_detection_model_insightscore > 900
   ```
3. In **Outcomes**, choose **Create a new outcome**. An outcome is the result from a fraud prediction and is returned if the rule matches during an evaluation.
4. In **Create a new outcome**, enter **`verify_customer`** as the outcome name. Optionally, enter a description.
5. Choose **Save outcome**. For details, see Create an outcome (p. 35).
6. Choose **Add rule** to run the rule validation checker and save the rule. After it's created, Amazon Fraud Detector makes the rule available for use in your detector.
7. Choose **Add another rule**, and then choose the **Create rule** tab.

8. Repeat this process twice more to create your `medium_fraud_risk` and `low_fraud_risk` rules using the following rule details:

   - medium_fraud_risk

     Rule name: `medium_fraud_risk`

     Outcome: `review`

     Expression:

     ```
     $sample_fraud_detection_model_insightscore <= 900 and

     $sample_fraud_detection_model_insightscore > 700
     ```

   - low_fraud_risk

     Rule name: `low_fraud_risk`

     Outcome: `approve`

     Expression:

     ```
     $sample_fraud_detection_model_insightscore <= 700
     ```

   These values are examples only. When creating rules for your own detector, you should use values that are appropriate based on your model, data and business.

9. After you have created all three rules, choose **Next**.

   For more information about creating and writing rules, see Create a rule (p. 26) and Rule language reference (p. 28).

Step 4: Configure rule execution and define rule order

The rule execution mode for the rules included in the detector version determines if all the rules you define are evaluated, or if rule evaluation stops at the first matched rule. You can define and edit the rule execution mode at the detector version level, when the detector version is in draft status.

The default rule execution mode is `FIRST_MATCHED`.

First matched

First matched rule execution mode returns the outcomes for the first matching rule based on defined rule order. If you specify `FIRST_MATCHED`, Amazon Fraud Detector evaluates rules sequentially, first to last, stopping at the first matched rule. Amazon Fraud Detector then provides the outcomes for that single rule.

The order in which you execute rules can affect the resulting fraud prediction outcome. After you have created your rules, re-order the rules to execute them in the desired order by following these steps:

If your `high_fraud_risk` rule is not already on the top of your rule list, choose **Order**, and then choose **1**. This moves `high_fraud_risk` to the first position.

Repeat this process so that your `medium_fraud_risk` rule is in the second position and your `low_fraud_risk` rule is in the third position.

All matched

All matched rule execution mode returns outcomes for all matched rules, regardless of rule order. If you specify `ALL_MATCHED`, Amazon Fraud Detector evaluates all rules and returns the outcomes for all matched rules.

Select `FIRST_MATCHED` and then choose **Next**.

Step 5: Review and create detector version

Next, review your detection logic and create the first version of your detector.

1. Review the detector details, models, and rules you have configured. If you need to make any changes, choose **Edit** next to the corresponding section.
2. Choose **Create detector**. After it's created, the first version of your detector appears in the Detector versions table with `Draft` status.

Step 6: Test and get predictions

In the Amazon Fraud Detector console, you can test your detector's logic using mock event data with the **Run test** feature.

1. Scroll to **Run test** at the bottom of the **Detector version details** page.
2. For **Event metadata**, enter a timestamp of when the event occurred, as well as a unique identifier for the entity performing the event. For this exercise, select a date from the date picker for timestamp, and enter "1234" for the Entity ID.
3. For **Event variable**, enter the variable values that you would like to test. For this exercise, you only need two input fields (that is, ip_address and email_address) because these are the inputs used to train your Amazon Fraud Detector model. You can use the following example values (assuming you used the suggested variable names):

   - ip_address: 205.251.233.178
   - email_address: johndoe@exampledomain.com
4. Choose **Run test**.
5. Amazon Fraud Detector returns the fraud prediction outcome based on the rule execution mode. If the rule execution mode is `FIRST_MATCHED`, then the returned outcome corresponds to the first rule (the highest priority) that matched (evaluated to true). If the rule execution mode is `ALL_MATCHED`, then the returned outcome corresponds to all rules that matched (evaluated to be true). Amazon Fraud Detector also returns the model score for any models added to your detector.
6. When you are satisfied that the detector is working as expected, you can promote it from `Draft` to `Active`, which makes the detector available for use in real-time fraud detection.

   On the **Detector version details** page, choose **Actions**, **Publish**, **Publish version**. This changes the detector's status from **Draft** to **Active.**

   At this point, your model and associated detector logic are ready to evaluate online activities for fraud in real-time using the Amazon Fraud Detector `GetEventPrediction` API. For a code sample, see Getting fraud predictions (p. 40).

Get started (AWS SDK for Python)

This topic explains how to get started programming Amazon Fraud Detector with the AWS SDK for Python (Boto3). For instruction on completing this exercise using the AWS Console, see Get started (console) (p. 6).

Prerequisites

The following are prerequisite steps for using the Python examples in this guide. Ensure you are using the Boto3 SDK version 1.14.29 or higher.

- Complete Setting up Amazon Fraud Detector (p. 4).
- Complete Get and upload example training data (p. 6) to upload the training data required for this exercise.

Step 1: Setup and verify your Python environment

Boto is the Amazon Web Services (AWS) SDK for Python. It enables Python developers to create, configure, and manage AWS services. For details on how to install Boto3, refer to the AWS SDK for Python (Boto3).

After installing, run the following Python example to confirm that your environment is configured correctly. If your environment is configured correctly, the response will contain a list of detectors. If no detectors have been created, the list will be empty.

```
import boto3
fraudDetector = boto3.client('frauddetector')

response = fraudDetector.get_detectors()
print(response)
```

Step 2: Create variables, entity type, and labels

After you verify that your Python environment is configured correctly, create the component resources that will be used in your events, models, and rules.

1. Create variables. A variable is a data element that can be used in models and rules. For a code sample, see Create a variable (p. 32).
2. Create an entity type. An entity type classifies who is performing the event, such as a customer or a merchant. For a code sample, see Create an entity type (p. 36).
3. Create labels. Labels are used to classify events as either fraudulent or legitimate. For a code sample, see Create a label (p. 37).

Step 3: Create event type

After you have created your component resources, you can create an event type. With Amazon Fraud Detector, you generate fraud predictions for events. An event type defines the structure for an individual

event sent to Amazon Fraud Detector. Once defined, you can build models and detectors that evaluate the risk for specific event types. For a code sample, see Create event types (p. 15).

Step 4: Create and train a model

Amazon Fraud Detector models learn to detect fraud for a specific event type. In Amazon Fraud Detector, you first create a model, which acts as a container for your model versions. Each time you train a model, a new version is created. For a code sample, see Building a model (p. 20).

The example code trains an Online Fraud Insights model using data stored in Amazon S3. Online Fraud Insights is a supervised machine learning model that can be adapted to detect a variety of online fraud and abuse risks such as new account fraud, online transaction fraud, and fraudulent reviews.

Once model training is complete, you can review model performance in the AWS Console or programmatically by calling the `DescribeModelVersions` API. To learn more, refer to Training performance metrics (p. 22).

After reviewing the model performance, activate the model to make it available for use by detectors in real-time fraud predictions. For a code sample, see Building a model (p. 20).

Step 5: Create a detector, outcome, rules, and detector version

A detector contains the detection logic, such as the models and rules, for a particular event that you want to evaluate for fraud. During a fraud prediction, you will specify the detector that you want to use to evaluate your event. To create a detector, complete the following steps.

1. Create a detector. A detector acts as a container for your detector versions. For a code sample, Create a detector (p. 25).
2. Create outcomes. An outcome is the result of a fraud prediction. For example, you may want outcomes to represent risk levels (high_risk, medium_risk, and low_risk). For a code sample, see Create an outcome (p. 35).
3. Create rules. A rule is a condition that tells Amazon Fraud Detector how to interpret variable values during a fraud prediction. A rule consists of one or more variables, a logic expression, and one or more outcomes. For a code sample, see Create a rule (p. 26).
4. Create a detector version. A detector version defines the specific models and rules that will be run as part of a fraud prediction. For a code sample, see Create a detector version (p. 25).

Step 6: Generate fraud predictions

To get fraud predictions, call the `GetEventPrediction` API. Supply information about the event you want to evaluate and synchronously receive a model score and outcome based on the designated detector. For a code sample, see Getting fraud predictions (p. 40).

(Optional) Explore the Amazon Fraud Detector APIs with a Jupyter (iPython) Notebook

For additional examples on how to use the Amazon Fraud Detector APIs, refer to the aws-fraud-detector-samples GitHub repository. Topics covered by the notebooks include building models and

detectors using the Amazon Fraud Detector APIs and making batch fraud prediction requests using the `GetEventPrediction` API.

Create event types

With Amazon Fraud Detector, you generate fraud predictions for events. An event type defines the structure for an individual event sent to Amazon Fraud Detector. Once defined, you can build models and detectors that evaluate the risk for specific event types.

The structure of an event includes the following:

- Entity Type: Classifies who is performing the event. During prediction, specify the entity type and entity Id to define who performed the event.
- Variables: Defines what variables can be sent as part of the event. Variables are used by models and rules to evaluate fraud risk. Once added, variables cannot be removed from an event type.
- Labels: Classifies an event as fraudulent or legitimate. Used during model training. Once added, labels cannot be removed form an event type.

For details on how to create an event type using the AWS Console see Step 1: Define event to evaluate for fraud (p. 7).

Create an event type using the AWS SDK for Python (Boto3)

The following example shows a sample request for the `PutEventType` API. The example assumes you have created the variables `ip_address` and `email_address`, the labels `legit` and `fraud`, and the entity type `sample_customer`. For information about how to create these resources, see Create resources (p. 32).

Note
You must first create variables, entity types, and labels prior to adding them to the event type.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.put_event_type (
name = 'sample_registration',
eventVariables = ['ip_address', 'email_address'],
labels = ['legit', 'fraud'],
entityTypes = ['sample_customer'])
```

Create model

A model version is the term Amazon Fraud Detector uses for a trained fraud detection machine learning model. All steps in the model training process are fully automated including data validation, data transformation, feature engineering, algorithm selection, training, and model optimization. Creating a model requires selecting the model type and specifying the model version configuration. The model type specifies the algorithms and transformations used to build the model.

Topics

Online fraud insights

Amazon Fraud Detector Online Fraud Insights is a supervised machine learning model designed to detect a variety of online fraud and risks. Because the model is supervised, it requires historical examples of fraudulent and legitimate events to train the model.

The Online Fraud Insights model uses an ensemble of machine-learning algorithms for data enrichment, transformation, and fraud classification. As part of the model training process, Online Fraud Insights enriches raw data elements like IP address and BIN number with 3rd party data such as the geo-location of the IP address or the issuing bank for a credit card. In addition to 3rd party data, Online Fraud Insights uses deep learning algorithms leveraging fraud patterns seen at Amazon and AWS. These fraud patterns become input features to your model using a gradient tree boosting algorithm.

To increase performance, Online Fraud Insights optimizes the hyper parameters of the gradient tree boosting algorithm via a Bayesian optimization process, sequentially training dozens of different models with varying model parameters (such as number of trees, depth of trees, number of samples per leaf) as well as different optimization strategies like upweighting the minority fraud population to take care of very low fraud rates.

By adapting the variables included in the model, Online Fraud Insights can be used to detect a variety of online fraud including:

- New account fraud: Accurately distinguish between legitimate and high-risk customer account registrations so you can selectively introduce additional steps or checks based on risk.
- Online payment fraud: Reduce online payment fraud by flagging suspicious online payment transactions before processing payments and fulfilling orders.
- Guest checkout fraud: Spot potential fraudsters among customers without transaction histories. You can send as little as two pieces of data from a guest checkout order (for example, email, IP address) to assess its potential fraud risk.
- Fake reviews abuse: Detect potentially fraudulent or fake reviews so that you can review prior to posting.

Preparing training data

Amazon Fraud Detector imports data only from files that are in the comma-separated values (CSV) format. Amazon Fraud Detector requires that the first row of your CSV file to contain column headers. The column headers in your CSV file need to map to the variables defined in the event type. For an example dataset, see Get and upload example training data (p. 6).

The Online Fraud Insights model requires a training dataset that has at least two variables, 10K total examples, and 400 examples of fraud to train a model. In addition to the event variables, the training dataset must contain the following headers:

- `EVENT_TIMESTAMP`: Defines when the event occurred. For more information, see Event Timestamps Format.
- `EVENT_LABEL`: Classifies the event as fraudulent or legitimate. The values in the column must correspond to the values defined in the event type.

For example, the following sample CSV data maps to the event type shown in Create event types (p. 15). This data represents historical registration events from an online merchant:

```
EVENT_TIMESTAMP,EVENT_LABEL,ip_address,email_address
4/10/2019 11:05,fraud,209.146.137.48,fake_burtonlinda@example.net
12/20/2018 20:04,legit,203.0.112.189,fake_davidbutler@example.org
3/14/2019 10:56,legit,169.255.33.54,fake_shelby76@example.net
1/3/2019 8:38,legit,192.119.44.26,fake_curtis40@example.com
9/25/2019 3:12,legit,192.169.85.29,fake_rmiranda@example.org
```

A simplified version of the corresponding event type is represented below. The event variables correspond to the headers in the CSV file and the values in `EVENT_LABEL` correspond to the values in the labels list.

```
(
name = 'sample_registration',
eventVariables = ['ip_address', 'email_address'],
labels = ['legit', 'fraud'],
entityTypes = ['sample_customer']
)
```

Data set guidance

The following guidance for training data will help you get the most out of your Online Fraud Insights model.

Gathering data

We recommend that you collect a minimum of six weeks of historic data, though three - six months of data is preferable. There is a maximum file size of 5GB. If the recommended six weeks of historic data exceeds this threshold, shorten the time range of your extraction.

Model variables

The Online Fraud Insights model requires at least two variables for model training. Generally, the more variables you provide the better the model can differentiate between fraud and legitimate events.

EVENT_TIMESTAMP

You must include the header `EVENT_TIMESTAMP`. Ensure your event timestamp is in the required format. For more information, see Timestamp formats (p. 18). As part of the model build process, the Online Fraud Insights model type orders your data based on the event timestamp, and splits your data for training and testing purposes. To get a fair estimate of performance, the model first trains on the training dataset, then tests this model on the test dataset.

EVENT_LABEL

You must include the header `EVENT_LABEL`. The Online Fraud Insights model requires a minimum of 400 observations are identified and labeled as “fraud”.

Data and label maturity

Ensure that records used to train the model have had sufficient time to “mature”, that is, enough time has passed to ensure “legitimate" and “fraud” records have been correctly identified. The maturity period is dependent on your business, and can take anywhere from two weeks to 90 days. For example, for chargeback fraud, it often takes 30 - 60 days (or more) to correctly identify fraudulent events. For the best model performance, ensure that all records in your training dataset are mature. For example, if your maturity is 30 days, ensure that the latest records in your dataset are at least 30 days old.

Sampling

The Online Fraud Insights training process will sample and partition historic data based on `EVENT_TIMESTAMP`. There is no need to manually sample the data and doing so may negatively impact your model results.

Nulls and missing values

Online Fraud Insights handles null and missing values. However, the percentage of nulls for variables should be limited. `EVENT_TIMESTAMP` and `EVENT_LABEL` columns should not contain any missing values.

Timestamp formats

Amazon Fraud Detector supports the following date/timestamp formats for the values in `EVENT_TIMESTAMP` during model training:

- %yyyy-%mm-%ddT%hh:%mm:%ssZ (ISO 8601 standard in UTC only with no milliseconds)

  Example: 2019-11-30T13:01:01Z
- %yyyy/%mm/%dd %hh:%mm:%ss (AM/PM)

  Examples: 2019/11/30 1:01:01 PM, or 2019/11/30 13:01:01
- %mm/%dd/%yyyy %hh:%mm:%ss

  Examples: 11/30/2019 1:01:01 PM, 11/30/2019 13:01:01
- %mm/%dd/%yy %hh:%mm:%ss

  Examples: 11/30/19 1:01:01 PM, 11/30/19 13:01:01

Amazon Fraud Detector makes the following assumptions when parsing date/timestamp formats for event timestamps:

- If you are using the ISO 8601 standard, it must be an exact match of the above specification
- If you are using one of the other formats, there is additional flexibility:
  - For months and days, you can provide single or double digits. For example, 1/12/2019 is a valid date.
  - You do not need to include hh:mm:ss if you do not have them (e.g. you can simply provide a date). You can also provide a subset of just the hour and minutes (e.g. hh:mm). Just providing hour is not supported. Milliseconds are also not supported.
  - If you provide AM/PM labels, a 12-hour clock is assumed. If there is no AM/PM information, a 24-hour clock is assumed.
  - You can use "/" or "-" as delimiters for the date elements. ":" is assumed for the timestamp elements.

Training data validations

As part of the training process, Online Fraud Insights will validate the training dataset for data quality issues that may impact model training. After validating the data, Amazon Fraud Detector will take appropriate action to build the best possible model. This includes issuing warnings for potential data quality issues, automatically removing variables that have data quality issues, or issuing an error and stopping the model training process.

Amazon Fraud Detector will fail to train a model if any of the following conditions are triggered:

- If the CSV is unable to be parsed
- If the datatype for a column is incorrect
- If the number of rows is < 10k
- If the number of rows identified as fraud or legitimate are < 400
- If more than 0.1% of values in `EVENT_TIMESTAMP` contains nulls or values other than the supported date/timestamp formats
- If more than 1% of the values in `EVENT_LABEL` contains nulls or values other than those defined in the event type
- If less than two variables are available for model training

Uploading to an Amazon S3 bucket

After you create a CSV file with your data, upload the file to your Amazon Simple Storage Service (Amazon S3) bucket.

To upload 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. Choose **Create bucket**.

   The **Create bucket** wizard opens.
3. In **Bucket name**, enter a DNS-compliant name for your bucket.

   The bucket name must:

   - Be unique across all of Amazon S3.
   - Be between 3 and 63 characters long.
   - Not contain uppercase characters.

- Start with a lowercase letter or number.

After you create the bucket, you can't change its name. For information about naming buckets, see Rules for Bucket Naming in the *Amazon Simple Storage Service Developer Guide*.

Important
Avoid including sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

4. In **Region**, choose the AWS Region where you want the bucket to reside. You must select the same Region in which you are using Amazon Fraud Detector, that is US East (N. Virginia), US East (Ohio), US West (Oregon), Europe (Ireland), Asia Pacific (Singapore) or Asia Pacific (Sydney).
5. In **Bucket settings for Block Public Access**, choose the Block Public Access settings that you want to apply to the bucket.

   We recommend that you leave all settings enabled. For more information about blocking public access, see Using Amazon S3 Block Public Access in the *Amazon Simple Storage Service Developer Guide*.
6. Choose **Create bucket**.
7. Upload training data file to your Amazon S3 bucket. Note the Amazon S3 location path for your training file (for example, s3://bucketname/object.csv).

Building a model

Amazon Fraud Detector models learn to detect fraud for a specific event type. In Amazon Fraud Detector, you first create a model, which acts as a container for your model versions. Each time you train a model, a new version is created. For details on how to create and train a model using the AWS Console see Step 2: Define model details (p. 7).

Each model has a corresponding model score variable. Amazon Fraud Detector creates this variable on your behalf when you create a model. You can use this variable in your rule expressions to interpret your model scores during a fraud evaluation.

Train and deploy a model using the AWS SDK for Python (Boto3)

A model version is created by calling the `CreateModel` and `CreateModelVersion` operations. `CreateModel` initiates the model, which acts as a container for your model versions. `CreateModelVersion` starts the training process, which results in a specific version of the model. A new version of the solution is created each time you call `CreateModelVersion`.

The following example shows a sample request for the `CreateModel` API. This example assumes you have created an event type `sample_registration`. For additional details about creating an event type, see Create event types (p. 15).

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.create_model (
modelId = 'sample_fraud_detection_model',
eventTypeName = 'sample_registration',
modelType = 'ONLINE_FRAUD_INSIGHTS')
```

Train your first version using the `CreateModelVersion` API. `TrainingDataSource` and `ExternalEventsDetail` specify the source and Amazon S3 location of the training data set.

`TrainingDataSchema` specifies how Amazon Fraud Detector should interpret the training data, specifically which event variables to include and how to classify the event labels.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.create_model_version (
modelId = 'sample_fraud_detection_model',
modelType = 'ONLINE_FRAUD_INSIGHTS',
trainingDataSource = 'EXTERNAL_EVENTS',
trainingDataSchema = {
    'modelVariables' : ['ip_address', 'email_address'],
    'labelSchema' : {
        'labelMapper' : {
            'FRAUD' : ['fraud'],
            'LEGIT' : ['legit']
        }
    }
},
externalEventsDetail = {
    'dataLocation' : 's3://bucket/file.csv',
    'dataAccessRoleArn' : 'role_arn'
}
)
```

A successful request will result in a new model version with status `TRAINING_IN_PROGRESS`. Once training is complete, the model version status will update to `TRAINING_COMPLETE`. You can review model performance using the Amazon Fraud Detector console or by calling `DescribeModelVersions`. For more information on how to interpret model scores and performance, see Model scores (p. 21) and Training performance metrics (p. 22).

After reviewing the model performance, activate the model to make it available to use by Detectors in real-time fraud predictions. Amazon Fraud Detector will deploy the model in multiple availability zones for redundancy with auto-scaling turned on to ensure the model scales with the number of fraud predictions you are making. To activate the model, call the `UpdateModelVersionStatus` API and update the status to `ACTIVE`.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.update_model_version_status (
modelId = 'sample_fraud_detection_model',
modelType = 'ONLINE_FRAUD_INSIGHTS',
modelVersionNumber = '1.00',
status = 'ACTIVE'
)
```

Model scores

Amazon Fraud Detector generates model scores between 0 and 1000, where 0 is low fraud risk and 1000 is high fraud risk. Model scores are directly related to the false positive rate (FPR). For example, a score of 600 corresponds to an estimated 10% false positive rate whereas a score of 900 corresponds to an estimated 2% false positive rate. The following table provides details of how certain model scores correlate to estimated false positive rates.

Model score	Estimated FPR
975	0.50%

Model score	Estimated FPR
950	1%
900	2%
860	3%
775	5%
700	7%
600	10%

Training performance metrics

After model training is complete, Amazon Fraud Detector validates model performance using 15% of your data that was not used to train the model. You can expect your trained Amazon Fraud Detector model to have real-world fraud detection performance that is similar to the validation performance metrics.

As a business, you must balance between detecting more fraud, and adding more friction to legitimate customers. To assist in choosing the right balance, Amazon Fraud Detector provides the following tools to assess model performance:

- **Score distribution chart** – A histogram of model score distributions assumes an example population of 100,000 events. The left Y axis represents the legitimate events and the right Y axis represents the fraud events. You can select a specific model threshold by clicking on the chart area. This will update the corresponding views in the confusion matrix and ROC chart.
- **Confusion matrix** – Summarizes the model accuracy for a given score threshold by comparing model predictions versus actual results. Amazon Fraud Detector assumes an example population of 100,000 events. The distribution of fraud and legitimate events simulates the fraud rate in your businesses.
  - **True positives** – The model predicts fraud and the event is actually fraud.
  - **False positives** – The model predicts fraud but the event is actually legitimate.
  - **True negatives** – The model predicts legitimate and the event is actually legitimate.
  - **False negatives** – The model predicts legitimate but the event is actually fraud.
  - **True positive rate (TPR)** – Percentage of total fraud the model detects. Also known as capture rate.
  - **False positive rate (FPR)** – Percentage of total legitimate events that are incorrectly predicted as fraud.
- **Receiver Operator Curve (ROC)** – Plots the true positive rate as a function of false positive rate over all possible model score thresholds. View this chart by choosing **Advanced Metrics**.
- **Area under the curve (AUC)** – Summarizes TPR and FPR across all possible model score thresholds. A model with no predictive power has an AUC of 0.5, whereas a perfect model has a score of 1.0.

To use the model performance metrics

1. Start with the **Score distribution** chart to review the distribution of model scores for your fraud and legitimate events. Ideally, you will have a clear separation between the fraud and legitimate events. This indicates the model can accurately identify which events are fraudulent and which are legitimate. Select a model threshold by clicking on the chart area. You can see how adjusting the model score threshold impacts your true positive and false positive rates.

Note
The score distribution chart plots the fraud and legitimate events on two different Y axis. The left Y axis represents the legitimate events and the right Y axis represents the fraud events.

2. Review the **Confusion matrix**. Depending on your selected model score threshold, you can see the simulated impact based on a sample of 100,000 events. The distribution of fraud and legitimate events simulates the fraud rate in your businesses. Use this information to find the right balance between true positive rate and false positive rate.
3. For additional details, choose **Advanced Metrics**. Use the ROC chart to understand the relationship between true positive rate and false positive rate for any model score threshold. The ROC curve can help you fine-tune the tradeoff between true positive rate and false positive rate.

   Note
   You can also review metrics in table form by choosing **Table**.
   The table view also shows the metric **Precision**. **Precision** is the percentage of fraud events correctly predicted as fraudulent as compared to all events predicted as fraudulent.
4. Use the performance metrics to determine the optimal model thresholds for your businesses based on your goals and fraud-detection use case. For example, if you plan to use the model to classify new account registrations as either high, medium, or low risk, you need to identify two threshold scores so you can draft three rule conditions as follows:

   - Scores > X are high risk
   - Scores < X but > Y are medium risk
   - Scores < Y are low risk

Import an SageMaker model

You can optionally import SageMaker-hosted models to Amazon Fraud Detector. Similar to Amazon Fraud Detector models, SageMaker models can be added to detectors and generate fraud predictions using the `GetEventPrediction` API. As part of the `GetEventPrediction` request, Amazon Fraud Detector will invoke your SageMaker endpoint and pass the results to your rules.

You can configure Amazon Fraud Detector to use the event variables sent as part of the `GetEventPrediction` request. If you choose to use event variables, you must provide an input template. Amazon Fraud Detector will use this template to transform your event variables into the required input payload to invoke the SageMaker endpoint. Alternatively, you can configure your SageMaker model to use a byteBuffer that is sent as part of the `GetEventPrediction` request.

Amazon Fraud Detector supports importing SageMaker algorithms that use JSON or CSV input formats and JSONLines or CSV output formats. Examples of supported SageMaker algorithms include XGBoost, Linear Learner, and Random Cut Forest.

Import an SageMaker model using the AWS SDK for Python (Boto3)

To import an SageMaker model, use the `PutExternalModel` API. The following example assumes the SageMaker endpoint `sagemaker-transaction-model` has been deployed, is `InService` status, and uses the XGBoost algorithm.

The input configuration specifies that Amazon Fraud Detector will use the event variables to construct the model input (`useEventVariables` is set to `TRUE`). The input format is TEXT_CSV, given XGBoost requires a CSV input. The csvInputTemplate specifies how to construct the CSV input from the variables sent as part of the `GetEventPrediction` request. This example assumes you have created the variables `order_amt`, `prev_amt`, `hist_amt` and `payment_type`.

The output configuration specifies the response format of the SageMaker model, and maps the appropriate CSV index to the Amazon Fraud Detector variable `sagemaker_output_score`. Once configured, you can use the output variable in rules.

Note
The output from an SageMaker model must be mapped to a variable with source `EXTERNAL_MODEL_SCORE`. You can't create these variables in the console using **Variables**. You must instead create them when you configure your model import.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.put_external_model (
modelSource = 'SAGEMAKER',
modelEndpoint = 'sagemaker-transaction-model',
invokeModelEndpointRoleArn = 'your_SagemakerExecutionRols_arn',
inputConfiguration = {
    'useEventVariables' : True,
    'eventTypeName' : 'sample_transaction',
    'format' : 'TEXT_CSV',
    'csvInputTemplate' : '{{order_amt}}, {{prev_amt}}, {{hist_amt}}, {{payment_type}}'
},

outputConfiguration = {
    'format' : 'TEXT_CSV',
    'csvIndexToVariableMap' : {
        '0' : 'sagemaker_output_score'
    }
},

modelEndpointStatus = 'ASSOCIATED'
)
```

Create a detector

A detector contains the detection logic, such as the models and rules, for a particular event that you want to evaluate for fraud. Each detector can evaluate one event type.

A detector can have multiple versions, with each version having a status of `DRAFT`, `ACTIVE`, or `INACTIVE`. Only one detector version can be in `ACTIVE` status at a time.

To create a detector using the AWS Console, see Step 1: Create a detector (p. 9).

Create a detector using the AWS SDK for Python (Boto3)

The following example shows a sample request for the `PutDetector` API. A detector acts as a container for your detector versions. The `PutDetector` API specifies what event type the detector will evaluate. The following example assumes you have created an event type `sample_registration`.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.put_detector (
detectorId = 'sample_detector',
eventTypeName = 'sample_registration'
)
```

After you create a detector, you can Create a rule (p. 26).

Create a detector version

A detector version defines the specific models and rules that will be run as part of the `GetEventPrediction` request. You can add any of the rules defined within a detector to the detector version. You can also add any model trained on the evaluated event type.

Each detector version has a status of `DRAFT`, `ACTIVE`, or `INACTIVE`. Only one detector version can be in `ACTIVE` status at a time. During the `GetEventPrediction` request, Amazon Fraud Detector will use the `ACTIVE` detector if no `DetectorVersion` is specified.

To create a detector using the AWS Console, see Step 1: Create a detector (p. 9).

Rule execution mode

Amazon Fraud Detector supports two different rule execution modes: `FIRST_MATCHED` and `ALL_MATCHED`.

- If the rule execution mode is `FIRST_MATCHED`, Amazon Fraud Detector evaluates rules sequentially, first to last, stopping at the first matched rule. Amazon Fraud Detector then provides the outcomes for that single rule. If a rule evaluates to false (not matched), the next rule in the list is evaluated.
- If the rule execution mode is `ALL_MATCHED`, then all rules in an evaluation are executed in parallel, regardless of their order. Amazon Fraud Detector executes all rules and returns the defined outcomes for every matched rule.

Create a detector version using the AWS SDK for Python (Boto3)

The following example shows a sample request for the `CreateDetectorVersion` API. The example assumes you have created the rules described in Create a rule (p. 26) and the model `sample_fraud_detection_model` as described in Building a model (p. 20). The rule execution mode is set to `FIRST_MATCHED`, therefore Amazon Fraud Detector will evaluate rules sequentially, first to last, stopping at the first matched rule. Amazon Fraud Detector then provides the outcomes for that single rule during the `GetEventPrediction response`.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.create_detector_version(
detectorId = 'sample_detector',
rules = [{
    'detectorId' : 'sample_detector',
    'ruleId' : 'high_fraud_risk',
    'ruleVersion' : '1'
},
{
    'detectorId' : 'sample_detector',
    'ruleId' : 'medium_fraud_risk',
    'ruleVersion' : '1'
},
{
    'detectorId' : 'sample_detector',
    'ruleId' : 'low_fraud_risk',
    'ruleVersion' : '1'
}
],
modelVersions = [{
    'modelId' : 'sample_fraud_detection_model',
    'modelType': 'ONLINE_FRAUD_INSIGHTS',
    'modelVersionNumber' : '1.00'
}],
ruleExecutionMode = 'FIRST_MATCHED'
)
```

To update the status of a detector version, use the `UpdateDetectorVersionStatus` API. The following example updates the detector version status from `DRAFT` to `ACTIVE`. During a `GetEventPrediction` request, if a detector ID is not specified, Amazon Fraud Detector will use the `ACTIVE` version of the detector.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.update_detector_version_status(
detectorId = 'sample_detector',
detectorVersionId = '1',
status = 'ACTIVE'
)
```

Create a rule

A rule is a condition that tells Amazon Fraud Detector how to interpret variable values during a fraud prediction. A rule consists of one or more variables, a logic expression, and one or more outcomes. A

detector must have at least one associated rule. Rules in a detector are evaluated as part of a fraud prediction.

Expressions

Each rule must contain a single expression that captures your business logic. All expressions must evaluate to a Boolean value (true or false) and be less than 4,000 characters in length. If-else type conditions are not supported. All variables used in the expression must be predefined in the evaluated event type.

To create a rule using the AWS Console, see Step 3: Add rules to a detector (p. 9).

Create a rule using the AWS SDK for Python (Boto3)

The following example creates three different rules (`high_fraud_risk`, `medium_fraud_risk`, and `low_fraud_risk`). The example assumes you have completed the steps outlined in Building a model (p. 20) and created the model `sample_fraud_detection_model`. The example also assumes you have created the outcomes `verify_customer`, `review`, and `approve` (see Create an outcome (p. 35) for details on how to create an outcome). The expressions compare the model score output variable `sample_fraud_detection_model_insightscore` against various thresholds to determine the level of risk for an event.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.create_rule(
ruleId = 'high_fraud_risk',
detectorId = 'sample_detector',
expression = '$sample_fraud_detection_model_insightscore > 900',
language = 'DETECTORPL',
outcomes = ['verify_customer']
)

fraudDetector.create_rule(
ruleId = 'medium_fraud_risk',
detectorId = 'sample_detector',
expression = '$sample_fraud_detection_model_insightscore <= 900 and
 $sample_fraud_detection_model_insightscore > 700',
language = 'DETECTORPL',
outcomes = ['review']
)

fraudDetector.create_rule(
ruleId = 'low_fraud_risk',
detectorId = 'sample_detector',
expression = '$sample_fraud_detection_model_insightscore <= 700',
language = 'DETECTORPL',
outcomes = ['approve']
)
```

After you create the rules, you can Create a detector version (p. 25).

You can update a rule by calling the `UpdateRuleVersion` API. The following example updates the model score thresholds for the rules `high_fraud_risk` and `medium_fraud_risk` from 900 to 950.

```
fraudDetector.update_rule_version(
rule = {
    'detectorId' : 'sample_detector',
    'ruleId' : 'high_fraud_risk',
```

```
    'ruleVersion' : '1'
},
expression = '$sample_fraud_detection_model_insightscore > 950',
language = 'DETECTORPL',
outcomes = ['verify_customer']
)

fraudDetector.update_rule_version(
rule = {
    'detectorId' : 'sample_detector',
    'ruleId' : 'medium_fraud_risk',
    'ruleVersion' : '1'
},
expression = '$sample_fraud_detection_model_insightscore <= 950 and
 $sample_fraud_detection_model_insightscore > 700',
language = 'DETECTORPL',
outcomes = ['review']
)
```

Rule language reference

The following section outlines the expression (that is, rule writing) capabilities in Amazon Fraud Detector.

Using variables

You can use any variable defined in the evaluated event type as part of your expression. Use the dollar sign to indicate a variable:

```
$example_variable < 100
```

Comparison, membership, and identity operators

Amazon Fraud Detector includes the following comparison operators: >, >=, <, <=,!=, ==, in, not in

The following are examples:

Example: <

```
$variable < 100
```

Example: in, not in

```
$variable in [5, 10, 25, 100]
```

Example: !=

```
$variable != "US"
```

Example: ==

```
$variable == 1000
```

Operator Tables

Operator	Amazon Fraud Detector Operator
Equal to	==
Not equal to	!=
Greater than	>
Less than	<
Great than or equal to	>=
Less than or equal to	<=
In	in
And	and
Or	or
Not	!

Basic math

You can use basic math operators in your expression (for example, +, -, * ,/). A typical use case is when you need to combine variables during your evaluation.

In the rule below, we are adding the variable `$variable_1` with `$variable_2`, and checking whether the total is less than 10.

```
$variable_1 + $variable_2 < 10
```

Basic Math Table Data

Operator	Amazon Fraud Detector Operator
Plus	+
Minus	-
Multiply	*
Divide	/
Modulo	%

Regular Expression (regex)

You can use regex to search for specific patterns as part of your expression. This is particularly useful if you are looking to match a specific string or numerical value for one of your variables. Amazon Fraud Detector only supports match when working with regular expressions (for example, it returns True/False depending on whether the provided string is matched by the regular expression). Amazon Fraud Detector's regular expression support is based on .matches() in java (using the RE2J Regular Expression library). There are several helpful websites on the internet that are useful for testing different regular expression patterns.

In the first example below, we first transform the variable `email` to lowercase. We then check whether the pattern `@gmail.com` is in the `email` variable. Notice the second period is escaped so that we can explicitly check for the string `.com`.

```
regex_match(".*@gmail\.com", lowercase($email))
```

In the second example, we check whether the variable `phone_number` contains the country code `+1` to determine if the phone number is from the US. The plus symbol is escaped so that we can explicitly check for the string `+1`.

```
regex_match(".*\+1", $phone_number)
```

Regex Table

Operator	Amazon Fraud Detector Example
Match any string that starts with	regex_match("^mystring", $variable)
Match entire string exactly	regex_match("mystring", $variable)
Match any character except new line	regex_match(".", $variable)
Match any number of characters except new line prior 'mystring'	regex_match(".*mystring", $variable)
Escape special characters	\

Checking for missing values

Sometimes it is beneficial to check whether the value is missing. In Amazon Fraud Detector this is represented by null. You can do this by using the following syntax:

```
$variable != null
```

Similarly, if you wanted to check whether a value is not present, you could do the following:

```
$variable == null
```

Multiple conditions

You can combine multiple expressions together using `and` and `or`. Amazon Fraud Detector stops in an `OR` expression when a single true value is found, and it stops in an `AND` when a single false value is found.

In the example below, we are checking for two conditions using the `and` condition. In the first statement, we are checking whether variable 1 is less than 100. In the second we check whether variable 2 is not the US.

Given the rule uses an `and`, both must be TRUE for the entire condition to evaluate to TRUE.

```
$variable_1 < 100 and $variable_2 != "US"
```

You can use parenthesis to group Boolean operations, as shown following:

```
$variable_1 < 100 and $variable_2 != "US" or ($variable_1 * 100.0 > $variable_3)
```

Other expression types

String Operators

Operator	Example
Transform string to uppercase	uppercase($variable)
Transform string to lowercase	lowercase($variable)

Other

Operator	Comment
Add a comment	# my comment

Create resources

Models, rules, and detectors use resources to evaluate events for fraud risk. This chapter provides information about creating resources.

Topics

Create a variable

Variables represent data elements that you want to use in a fraud prediction, such as data from the event that is being evaluated or risk score outputs from Amazon Fraud Detector models or Amazon SageMaker models.

To create a variable using the AWS Console, navigate to Amazon Fraud Detector, choose `Variables` in the left navigation, then choose **Create**.

You can optionally assign variables a variable type. Variable types represent common data elements used during fraud predictions. Only variables with an associated variable type can be used for model training. Select from a number of pre-defined variable types or one of three custom variable types `FREE_FORM_TEXT`, `CATEGORICAL`, or `NUMERIC`.

Variables must have a data type for the data element that the variable represents. For variables that are mapped to a variable type, the data type is pre-selected. Possible data types include:

Data type	Description	Default value	Example values
String	Any combination of letters and/or whole numbers	<empty>	abc, 123, 1D3B
Integer	Positive or negative whole numbers	0	1, -1
Boolean	True or False	False	True, False
Float	Numbers with decimal points	0.0	4.01, 0.10

Variables must have a default value. During a `GetEventPrediction` API call, this value will be used to run a rule or model if Amazon Fraud Detector does not receive a value for a variable. Default values must match the selected data type. In the AWS Console, Amazon Fraud Detector will automatically assign the default value of `0` for integers, `false` for Booleans, `0.0` for floats, and (empty) for strings. You can optionally set a custom default value.

Variable types

Amazon Fraud Detector supports the following variable types:

Cat	Variable type	Description	Data type	Exa
Email	EMAIL_ADDRESS	Email address collected during the event	String	abc@domain.com
IP address	IP_ADDRESS	IP address collected during the event	String	1.1.1.1
Phone number	PHONE_NUMBER	Phone number collected during the event	String	1-123-456-7891
Browser/ Device	USERAGENT	User agent collected during the event	String	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100101
	FINGERPRINT	Unique identifier for a device	String	sadfow987u234
Payment Instrument	PAYMENT_TYPE	Payment instrument type used for payment during the event	String	Credit Card
	CARD_BIN	First six digits of the credit card	Integer	123456
	AUTH_CODE	Alphanumerical code sent by a credit card issuer or issuing bank	String	00
	AVS	Address Verification System response code from card processor	String	Y
Billing Address	BILLING_NAME	Name associated with billing address	String	John Doe
	BILLING_PHONE	Phone associated with billing address	String	1-123-456-7891
	BILLING_ADDRESS_L1	Billing address line 1	String	123 4th St.
	BILLING_ADDRESS_L2	Billing address line 2	String	Unit 123
	BILLING_CITY	Billing address city	String	Seattle

Cat	Variable type	Description	Data type	Exa
	BILLING_STATE	Billing address state or providence	String	WA
	BILLING_COUNTRY	Billing address country	String	US
	BILLING_ZIP	Billing address postal code	String	98109
Shipping Address	SHIPPING_NAME	Name associated with shipping address	String	John Doe
	SHIPPING_PHONE	Phone associated with shipping address	String	1-123-456-7891
	SHIPPING_ADDRESS_L1	Shipping address line 1	String	123 4th St.
	SHIPPING_ADDRESS_L2	Shipping address line 2	String	Unit 123
	SHIPPING_CITY	Shipping address city	String	Seattle
	SHIPPING_STATE	Shipping address state or providence	String	WA
	SHIPPING_COUNTRY	Billing address country	String	US
	SHIPPING_ZIP	Shipping address postal code	String	98109
Order	ORDER_ID	Unique identifier for transaction	String	LUX60
	PRODUCT_CATEGORY	Product category of order item	String	kitchen
	CURRENCY_CODE	ISO 4217 currency code	String	USD
	PRICE	Total order price	String	560.00
Custom	NUMERIC	Any variable that can be represented as a real number	Float	1.224
	CATEGORICAL	Any variable that describes categories, segments, or groups	String	Large

Cat	Variable type	Description	Data type	Exa
	FREE_FORM_TEXT	Any free form text that is captured as part of the event. For example, a customer review or comment.	String	Example of a free form text input

Create a variable using the AWS SDK for Python (Boto3)

The following example shows requests for the `CreateVariable` API. The example creates two variables, `email_address` and `ip_address`, and assigns them to the corresponding variable types (`EMAIL_ADDRESS` and `IP_ADDRESS`). Specifying the variable type allows Amazon Fraud Detector to interpret the variable during model training and `GetEventPrediction` requests. Only variables with an associated variable type can be used for model training.

You must specify a variable source, which identifies where the variable value will be derived. If the variable source is **EVENT**, the variable value will be sent as part of the `GetEventPrediction` request. If the variable value is `MODEL_SCORE`, it will be populated by an Amazon Fraud Detector model. If `EXTERNAL_MODEL_SCORE`, the variable value will be populated by an imported SageMaker model.

```
import boto3
fraudDetector = boto3.client('frauddetector')

 #Create variable email_address
fraudDetector.create_variable(
name = 'email_address',
variableType = 'EMAIL_ADDRESS',
dataSource = 'EVENT',
dataType = 'STRING',
defaultValue = '<unknown>'
)

#Create variable ip_address
fraudDetector.create_variable(
name = 'ip_address',
variableType = 'IP_ADDRESS',
dataSource = 'EVENT',
dataType = 'STRING',
defaultValue = '<unknown>'
)
```

Create an outcome

An outcome is the result of a fraud prediction. Create an outcome for each possible fraud prediction result. For example, you may want outcomes to represent risk levels (high_risk, medium_risk, and low_risk) or actions (approve, review). Once created, you can add one or more outcomes to a rule. As part of the `GetEventPrediction` response, Amazon Fraud Detector will return the defined outcomes for any matched rule.

To create a outcome using the AWS Console, open the AWS Console and sign in to your account. Navigate to Amazon Fraud Detector and in the left navigation pane, choose **Outcomes**, then choose **Create**.

Create an outcome using the AWS SDK for Python (Boto3)

The following example shows requests for the `CreateOutcome` API. The example creates three outcomes, `verify_customer`, `review`, and `approve`. You can then assign these outcomes to rules.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.put_outcome(
name = 'verify_customer',
description = 'this outcome initiates a verification workflow'
)

fraudDetector.put_outcome(
name = 'review',
description = 'this outcome sidelines event for review'
)

fraudDetector.put_outcome(
name = 'approve',
description = 'this outcome approves the event'
)
```

Create an entity type

An entity represents who is performing the event. As part of a fraud prediction, you can pass the entity ID to indicate the specific entity who performed the event.

An entity type classifies the entity. Example classifications include customer, merchant, or account.

To create a entity type using the AWS Console, open the AWS Console and sign in to your account. Navigate to Amazon Fraud Detector and in the left navigation pane, choose **Entities**, then choose **Create**.

Create an entity type using the AWS SDK for Python (Boto3)

The following example shows a request for the `PutEntityType` API. The example creates the entity type `sample_customer`. You can then associate this entity with an event type to describe who is performing the event.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.put_entity_type(
name = 'sample_customer',
description = 'sample customer entity type'
)
```

Create a label

A label classifies an event as fraudulent or legitimate. Once you have created a label, add the label to the event type by calling the PutEventType API.

Use labels to train supervised machine learning models in Amazon Fraud Detector. As part of model version configuration, you classify the event labels as either fraudulent or legitimate. The supervised model will then learn to classify events using the variable values.

To create a label using the AWS Console, open the AWS Console and sign in to your account. Navigate to Amazon Fraud Detector and in the left navigation pane, choose **Labels**, then choose **Create**.

Create a label using the AWS SDK for Python (Boto3)

The following example shows requests for the `PutLabel` API. The example creates two labels (fraud, legit). You can add these labels to an event type to classify specific events.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.put_label(
name = 'fraud',
description = 'label for fraud events'
)

fraudDetector.put_label(
name = 'legit',
description = 'label for legitimate events'
)
```

Delete resources

Models, rules, and detectors use resources to evaluate events for fraud risk. This chapter provides information about deleting resources.

Topics

Delete a detector, detector version, or rule version

Before deleting a detector, you must first delete all detector versions and rule versions associated with the detector.

To delete a detector version

You can only delete detector versions that are in `DRAFT` or `INACTIVE` status.

1. In the left navigation pane of the Amazon Fraud Detector console, choose **Detectors**.
2. Choose the detector that contains the detector version you want to delete.
3. Choose the detector version you want to delete.
4. Choose **Actions**, then choose **Delete**.
5. Type `delete` and then choose **Delete detector**.

To delete a rule version

You can delete a rule version only if it is not used by any `ACTIVE` or `INACTIVE` detector versions. If necessary, before deleting a rule version, first move the `ACTIVE` detector version to `INACTIVE`, then delete the `INACTIVE` detector version.

1. In the left navigation pane of the Amazon Fraud Detector console, choose **Detectors**.
2. Choose the detector that contains the rule version you want to delete.
3. Choose the **Associated rules** tab and choose the rule you want to delete.
4. Choose the rule version you want to delete.
5. Choose **Actions**, then choose **Delete rule version**.
6. Type `delete` and then choose **Delete version**.

To delete a detector

Before deleting a detector, you must first delete all detector versions and rule versions associated with the detector.

1. In the left navigation pane of the Amazon Fraud Detector console, choose **Detectors**.
2. Choose the detector you want to delete.
3. Choose **Actions**, then choose **Delete detector**.
4. Type `delete` and then choose **Delete detector**.

Delete an event

When you delete an event, Amazon Fraud Detector permanently deletes that event from the evaluation history and the event data is no longer stored in Amazon Fraud Detector.

To delete an event

1. In the left navigation pane of the Amazon Fraud Detector console, choose **Search past predictions**.
2. Choose the event that you want to delete.
3. Choose **Actions**, then choose **Delete event**.
4. Type `delete` and then choose **Delete event**.

Getting fraud predictions

To get fraud predictions, call the `GetEventPrediction` API. Supply information about the event you want to evaluate and synchronously receive a model score and outcome based on the designated detector.

As part of the request you must specify the `detectorId` that Amazon Fraud Detector will use to evaluate the event. You can optionally specify a `detectorVersionId`. If a `detectorVersionId` is not specified, Amazon Fraud Detector will use the `ACTIVE` version of the detector.

You are required to provide the following metadata regarding the evaluated event:

- EventId: A unique identifier for the event.
- Entities: The `entityType` and `entityId` to specify who is performing the event. If the `entityId` is not available at the time of evaluation, pass the string `unknown`.
- Timestamp: The timestamp when the event occurred. The timestamp must be in ISO 8601 standard in UTC.
- Event variables: Names of the event type's variables and their corresponding values for the evaluated event. If you do not send a value for a variable, Amazon Fraud Detector will use the default value.

You can optionally send data to invoke an SageMaker model by passing the data in the field `externalModelEndpointBlobs`.

During the evaluation, Amazon Fraud Detector will first generate model scores for any models added to the detector version, then pass the results to the rules for evaluation. The rules will be executed as specified by the rule execution mode (see Create a detector version (p. 25) for details). As part of the response, Amazon Fraud Detector will provide model scores as well as any outcomes associated to the matched rules.

To generate a test prediction using the AWS Console, see Step 6: Test and get predictions (p. 11)

Get a fraud prediction using the AWS SDK for Python (Boto3)

To generate a fraud prediction, call the `GetEventPrediction` API. The example below assumes you have completed Part B: Generate real-time fraud predictions (p. 8). As part of the response, you will receive a model score as well as any matched rules and corresponding outcomes. You can find additional examples of `GetEventPrediction` requests on the aws-fraud-detector-samples GitHub repository.

```
import boto3
fraudDetector = boto3.client('frauddetector')

fraudDetector.get_event_prediction(
detectorId = 'sample_detector',
eventId = '802454d3-f7d8-482d-97e8-c4b6db9a0428',
eventTypeName = 'sample_registration',
eventTimestamp = '2020-07-13T23:18:21Z',
entities = [{'entityType':'sample_customer', 'entityId':'12345'}],
eventVariables = {
    'email_address' : 'johndoe@exampledomain.com',
    'ip_address' : '1.2.3.4'
```

```
}
)
```

Security in Amazon Fraud Detector

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Fraud Detector, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Fraud Detector. The following topics show you how to configure Amazon Fraud Detector to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Fraud Detector resources.

Topics

Data Protection in Amazon Fraud Detector

Amazon Fraud Detector conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.

- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon Fraud Detector or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Fraud Detector or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Encryption at rest

Amazon Fraud Detector encrypts your data at rest with your choice of an encryption key. You can choose one of the following:

- An AWS owned customer master key (CMK). If you don't specify an encryption key your data is encrypted with this key by default.
- A customer managed CMK. You can provide the ARN of an encryption key that you created in your account. When you use a customer managed CMK, you must give the key a key policy that enables Amazon Fraud Detector to use the key. Select a symmetric customer managed CMK. Amazon Fraud Detector does not support asymmetric CMKs. For more information, see Key management (p. 43).

Encryption in transit

Amazon Fraud Detector copies data out of your account and processes it in an internal AWS system. By default, Amazon Fraud Detector uses TLS 1.2 with AWS certificates to encrypt data in transit.

Key management

Amazon Fraud Detector encrypts your data using one of two types of keys:

- An AWS owned customer master key (CMK). This is the default.
- A customer managed CMK. You can create the key using the AWS KMS console. Select a symmetric customer managed CMK, Amazon Fraud Detector does not support asymmetric CMKs. For more information, see Using Symmetric and Asymmetric Keys in the AWS Key Management Service Developer Guide.

When you create a key using the AWS KMS console, you must give the key the following policy that enables Amazon Fraud Detector to use the key. For more information, see Using Key Policies in AWS KMS in the AWS Key Management Service Developer Guide.

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "frauddetector.amazonaws.com"
    },
    "Action": [
        "kms:Encrypt",
```

```
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey",
        "kms:CreateGrant",
        "kms:RetireGrant"
    ],
    "Resource": "*"
}
```

To use a customer managed CMK to encrypt your Amazon Fraud Detector data, use Amazon Fraud Detector's PutKMSEncryptionKey API.

Note
When you use customer managed CMK to encrypt your Amazon Fraud Detector data, the data encrypted using this method is not searchable using filters in the Search Past Predictions area of the Amazon Fraud Detector console. To ensure complete search results, use one or more of the following properties to filter results:

- Event ID
- Evaluation timestamp
- Detector status
- Detector version
- Model version
- Model type
- Rule evaluation status
- Rule execution mode
- Rule match status
- Rule version
- Variable data source

Amazon Fraud Detector and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon Fraud Detector by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access Amazon Fraud Detector APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with Amazon Fraud Detector APIs. Traffic between your VPC and Amazon Fraud Detector does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Considerations for Amazon Fraud Detector VPC endpoints

Before you set up an interface VPC endpoint for Amazon Fraud Detector, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

Amazon Fraud Detector supports making calls to all of its API actions from your VPC.

VPC endpoint policies are supported for Amazon Fraud Detector. By default, full access to Amazon Fraud Detector is allowed through the endpoint. For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Creating an interface VPC endpoint for Amazon Fraud Detector

You can create a VPC endpoint for the Amazon Fraud Detector service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create a VPC endpoint for Amazon Fraud Detector using the following service name:

- com.amazonaws.*region*.frauddetector

If you enable private DNS for the endpoint, you can make API requests to Amazon Fraud Detector using its default DNS name for the Region, for example, `frauddetector.us-east-1.amazonaws.com`.

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for Amazon Fraud Detector

You can create a policy for interface VPC endpoints for Amazon Fraud Detector to specify the following:

- The principal that can perform actions
- The actions that can be performed
- The resources on which actions can be performed

For more information, see Controlling Access to Services with VPC Endpoints in the *Amazon VPC User Guide*.

The following example VPC endpoint policy specifies that all users who have access to the VPC interface endpoint are allowed to access the Amazon Fraud Detector detector named `my_detector`.

```
{
  "Statement": [
      {
          "Action": "frauddetector:*Detector",
          "Effect": "Allow",
          "Resource": "arn:aws:frauddetector:us-east-1:123456789012:detector/my_detector",
          "Principal": "*"
      }
  ]
}
```

In this example, the following are denied:

- Other Amazon Fraud Detector API actions
- Invoking Amazon Fraud Detector `GetEventPrediction` API

> **Note**
> In this example, users can still take other Amazon Fraud Detector API actions from outside the VPC. For information about how to restrict API calls to those from within the VPC, see Amazon Fraud Detector identity-based policies (p. 50).

Opting out of using your data for service improvement

Historical event data you provide to train models and generate predictions is used solely to provide and maintain the service and, unless you opt out as provided below, this data may be used to improve the quality of Amazon Fraud Detector and other AWS fraud prevention services. Your trust, privacy, and the security of your content are our highest priority and we implement appropriate and sophisticated technical and physical controls, including encryption at rest and in transit, designed to prevent unauthorized access to, or disclosure of, your content and ensure that our use complies with our commitments to you. See Data Privacy FAQ for more information. You may opt out of having your data used to develop or improve the quality of Amazon Fraud Detector and other AWS fraud prevention services using the following process:

1. Create an Account and billing support case using the AWS Support Center in the AWS Console.
2. Select **Account** for **Type**.
3. Select **Security** for **Category**.
4. Enter "Opt out of having my data used to improve Amazon Fraud Detector" in **Subject**.
5. Enter a message such as "I would like to opt out of having my data used to improve the quality of Amazon Fraud Detector and other AWS fraud prevention services." for **Description**.
6. Select your preferred contact option, and choose **Submit**.

A support case will be created that you can use to track your opt-out request. Response times will vary depending on your support plan.

Identity and access management for Amazon Fraud Detector

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Fraud Detector resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Amazon Fraud Detector.

Service user – If you use the Amazon Fraud Detector service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Fraud

Detector features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Fraud Detector, see Troubleshooting Amazon Fraud Detector identity and access (p. 57).

Service administrator – If you're in charge of Amazon Fraud Detector resources at your company, you probably have full access to Amazon Fraud Detector. It's your job to determine which Amazon Fraud Detector features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Fraud Detector, see How Amazon Fraud Detector works with IAM (p. 50).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Fraud Detector. To view example Amazon Fraud Detector identity-based policies that you can use in IAM, see Amazon Fraud Detector identity-based policy examples (p. 53).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

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

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM Users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key

pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

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

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

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the *IAM User Guide*.

To learn whether to use IAM roles, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

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

Identity-based policies

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

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

Access Control Lists (ACLs)

Access control lists (ACLs) are a type of policy that controls which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify

the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs Work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the *IAM User Guide*.

Multiple policy types

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

How Amazon Fraud Detector works with IAM

Before you use IAM to manage access to Amazon Fraud Detector, you should understand what IAM features are available to use with Amazon Fraud Detector. To get a high-level view of how Amazon Fraud Detector and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

Amazon Fraud Detector identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon Fraud Detector supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

To get started with Amazon Fraud Detector, we recommend creating an IAM user with access restricted to Amazon Fraud Detector operations and required permissions. You can add other permissions as needed. The following policies provide the required permission to use Amazon Fraud Detector: `AmazonFraudDetectorFullAccessPolicy` and `AmazonS3FullAccess`. For more information on setting up Amazon Fraud Detector using these policies see Setting up Amazon Fraud Detector (p. 4).

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon Fraud Detector use the following prefix before the action: `frauddetector:`. For example, to create a rule with the Amazon Fraud Detector `CreateRule` API operation, you include the `frauddetector:CreateRule` action in the policy. Policy statements must include either an `Action` or `NotAction` element. Amazon Fraud Detector defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "frauddetector:action1",
      "frauddetector:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "frauddetector:Describe*"
```

To see a list of Amazon Fraud Detector actions, see Actions Defined by Amazon Fraud Detector in the *IAM User Guide*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

Resource Types Defined by Amazon Fraud Detector lists all Amazon Fraud Detector resource ARNs.

For example, to specify the `my_detector` detector in your statement, use the following ARN:

```
"Resource": "arn:aws:frauddetector:us-east-1:123456789012:detector/my_detector"
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

To specify all detectors that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:frauddetector:us-east-1:123456789012:detector/*"
```

Some Amazon Fraud Detector actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

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

To see a list of Amazon Fraud Detector resource types and their ARNs, see Resources Defined by Amazon Fraud Detector in the *IAM User Guide*. To learn which actions you can specify the ARN of each resource, see Actions Defined by Amazon Fraud Detector.

Condition keys

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

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single

condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

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

Amazon Fraud Detector defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

To see a list of Amazon Fraud Detector condition keys, see Condition Keys for Amazon Fraud Detector in the *IAM User Guide*. To learn which actions and resources you can use a condition key, see Actions Defined by Amazon Fraud Detector.

Examples

To view examples of Amazon Fraud Detector identity-based policies, see Amazon Fraud Detector identity-based policy examples (p. 53).

Amazon Fraud Detector resource-based policies

Amazon Fraud Detector does not support resource-based policies.

Authorization Based on Amazon Fraud Detector Tags

You can attach tags to Amazon Fraud Detector resources or pass tags in a request to Amazon Fraud Detector. To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

Amazon Fraud Detector IAM roles

An IAM role is an entity within your AWS account that has specific permissions.

Using temporary credentials with Amazon Fraud Detector

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Amazon Fraud Detector supports using temporary credentials.

Service-linked roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Amazon Fraud Detector does not support service-linked roles.

Service roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Amazon Fraud Detector supports service roles.

Amazon Fraud Detector identity-based policy examples

By default, IAM users and roles don't have permission to create or modify Amazon Fraud Detector resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

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

Topics

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon Fraud Detector resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using Amazon Fraud Detector quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

AWS-managed (predefined) policy for Amazon Fraud Detector

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases

so that you can avoid having to investigate which permissions are needed. For more information, see AWS Managed Policies in the IAM User Guide.

The following AWS managed policy, which you can attach to users in your account, is specific to Amazon Fraud Detector:

`AmazonFraudDetectorFullAccess`: Grants full access to Amazon Fraud Detector resources, actions and the supported operations including:

- List and describe all model endpoints in Amazon SageMaker
- List all IAM roles in the account
- List all Amazon S3 buckets
- Allow IAM Pass Role to pass a role to Amazon Fraud Detector

This policy does not provide unrestricted S3 access. If you need to upload model training datasets to S3, the `AmazonS3FullAccess` managed policy (or scoped-down custom Amazon S3 access policy) is also required.

You can review the policy's permissions by signing in to the IAM console and searching by the policy name. You can also create your own custom IAM policies to allow permissions for Amazon Fraud Detector actions and resources as you need them. You can attach these custom policies to the IAM users or groups that require them.

Allow users to view their own permissions

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

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

```
}
```

Allowing full access to Amazon Fraud Detector resources

The following example gives an IAM user in your AWS account full access to all Amazon Fraud Detector resources and actions.

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

Allowing read-only access to Amazon Fraud Detector resources

In this example, you grant an IAM user in your AWS account read-only access to your Amazon Fraud Detector resources.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
          "frauddetector:GetEventTypes",
                "frauddetector:BatchGetVariable",
                "frauddetector:DescribeDetector",
                "frauddetector:GetModelVersion",
                "frauddetector:GetEventPrediction",
                "frauddetector:GetExternalModels",
                "frauddetector:GetLabels",
                "frauddetector:GetVariables",
                "frauddetector:GetDetectors",
                "frauddetector:GetRules",
                "frauddetector:ListTagsForResource",
                "frauddetector:GetKMSEncryptionKey",
                "frauddetector:DescribeModelVersions",
                "frauddetector:GetDetectorVersion",
                "frauddetector:GetPrediction",
                "frauddetector:GetOutcomes",
                "frauddetector:GetEntityTypes",
                "frauddetector:GetModels"
            ],
            "Resource": "*"
        }
    ]
}
```

Allowing access to a specific resource

In this example of a resource-level policy, you grant an IAM user in your AWS account access to all actions and resources except for one particular Detector resource.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "frauddetector:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "frauddetector:*Detector"
            ],
            "Resource": "arn:${Partition}:frauddetector:${Region}:${Account}:detector/
${detector-name}"
        }
    ]
}
```

Limiting access based on tags

This example policy demonstrates how to limit access to Amazon Fraud Detector based on resource tags. This example assumes that:

- In your AWS account you have defined two different IAM groups, named Team1 and Team2
- You have created four detectors
- You want to allow members of Team1 to make API calls on 2 detectors
- You want to allow members of Team2 to make API calls on the other 2 detectors

To control access to API calls (example)

1. Add a tag with the key `Project` and value `A` to the detectors used by Team1.
2. Add a tag with the key `Project` and value `B` to the detectors used by Team2.
3. Create an IAM policy with a `ResourceTag` condition that denies access to detectors that have tags with key `Project` and value `B`, and attach that policy to Team1.
4. Create an IAM policy with a `ResourceTag` condition that denies access to detectors that have tags with key `Project` and value `A`, and attach that policy to Team2.

The following is an example of a policy that denies all API calls on any Amazon Fraud Detector resource that has a tag with a key of `Project` and a value of `B`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "frauddetector:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "frauddetector:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/Project": "B"
        }
      }
    }
  ]
}
```

You can modify the scope of this policy by listing the specific action or actions that will be allowed or denied by the policy in the `Action` element.

Troubleshooting Amazon Fraud Detector identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Fraud Detector and IAM.

Topics

I am not authorized to perform an action in Amazon Fraud Detector

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a *`detector`* but does not have `frauddetector:`*`GetDetectors`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 frauddetector:GetDetectors on resource: my-example-detector
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-detector`* resource using the `frauddetector:`*`GetDetectors`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Amazon Fraud Detector.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Fraud Detector. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access Amazon Fraud Detector

To allow others to access Amazon Fraud Detector, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon Fraud Detector.

To get started right away, see Creating Your First IAM Delegated User and Group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my Amazon Fraud Detector resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Fraud Detector supports these features, see How Amazon Fraud Detector works with IAM (p. 50).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing Access to an IAM User in Another AWS Account That You Own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing Access to AWS Accounts Owned by Third Parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing Access to Externally Authenticated Users (Identity Federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.

Amazon Fraud Detector could not assume the given role

If you receive an error that Amazon Fraud Detector could not assume the given role, then you must update the trust relationship for the specified role. By specifying Amazon Fraud Detector as a trusted entity, the service can assume the role. When you use Amazon Fraud Detector to create a role, this trust relationship is automatically set. You only need to establish this trust relationship for IAM roles that are not created by Amazon Fraud Detector.

To establish a trust relationship for an existing role to Amazon Fraud Detector

1. Open the IAM console at https://console.aws.amazon.com/iam/
2. In the navigation pane choose **Roles**.
3. Choose the name of the role that you want to modify, and choose the **Trust relationships** tab.
4. Choose **Edit trust relationship**.
5. Under **Policy Document**, paste the following, and then choose **Update Trust Policy**.

```
{
    "Version": "2012-10-17",
    "Statement": [ {
        "Effect": "Allow",
        "Principal": {
            "Service": "frauddetector.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    } ]
}
```

Logging and monitoring in Amazon Fraud Detector

AWS provides the following monitoring tools to watch Amazon Fraud Detector, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. For more information on CloudWatch, see the Amazon CloudWatch User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. For more information an CloudTrail, see the AWS CloudTrail User Guide.

For more information on monitoring Amazon Fraud Detector, see Monitoring Amazon Fraud Detector (p. 61).

Compliance Validation for Amazon Fraud Detector

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

  Note
  Not all services are compliant with HIPAA.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Fraud Detector

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in Amazon Fraud Detector

As a managed service, Amazon Fraud Detector is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Amazon Fraud Detector through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Monitoring Amazon Fraud Detector

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Fraud Detector and your other AWS solutions. AWS provides the following monitoring tools to watch Amazon Fraud Detector, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For more information, see the Amazon CloudWatch User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Topics

Monitoring Amazon Fraud Detector with Amazon CloudWatch

You can monitor Amazon Fraud Detector using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

Topics

Using CloudWatch Metrics for Amazon Fraud Detector

To use metrics, you must specify the following information:

- The metric dimension or no dimension. A *dimension* is a name-value pair that helps you to uniquely identify a metric.
- The metric name, such as `GetEventPrediction`.

You can get monitoring data for Amazon Fraud Detector by using the AWS Management Console, the AWS CLI, or the CloudWatch API. You can also use the CloudWatch API through one of the Amazon AWS Software Development Kits (SDKs) or the CloudWatch API tools. The console displays a series of graphs based on the raw data from the CloudWatch API. Depending on your needs, you might prefer to use either the graphs displayed in the console or retrieved from the API.

The following list shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list.

How Do I?	Relevant Metrics
How do I track the number of predictions that have been performed?	Monitor the `GetEventPrediction` metric.
How can I monitor `GetEventPrediction` errors?	Use the `GetEventPrediction5xxError` and the `GetEventPrediction4xxError` metrics.
How can I monitor the latency of `GetEventPrediction` calls?	Use the `GetEventPredictionLatency` metric.

You must have the appropriate CloudWatch permissions to monitor Amazon Fraud Detector with CloudWatch. For more information, see Authentication and Access Control for Amazon CloudWatch.

Access Amazon Fraud Detector Metrics

The following steps show how to access Amazon Fraud Detector metrics using the CloudWatch console.

To view metrics (console)

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Metrics**, choose the **All Metrics** tab, and then choose **Fraud Detector**.
3. Choose the metric dimension.
4. Choose the desired metric from the list, and choose a time period for the graph.

Create an Alarm

You can create a CloudWatch alarm that sends an Amazon Simple Notification Service (Amazon SNS) message when the alarm changes state. An alarm watches a single metric over a time period that you specify. It performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or an Auto Scaling policy.

Alarms invoke actions for sustained state changes only. CloudWatch alarms don't invoke actions simply because they are in a particular state. The state must have changed and have been maintained for a specified number of time periods.

To set an alarm (console)

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, and choose **Create Alarm**. This opens the **Create Alarm Wizard**.
3. Choose **Select metric**.
4. In the **All metrics** tab, choose **Fraud Detector**.
5. Choose **By Detector ID**, and then choose the **GetEventPrediction** metric.
6. Choose the **Graphed metrics** tab.
7. For **Statistic**, choose **Sum**.
8. Choose **Select metric**.

9. For **Conditions**, choose **Static** for **Threshold type** and **Greater** for **Whenever...**, and then enter a maximum value of your choice. Choose **Next**.
10. To send alarms to an existing Amazon SNS topic, for **Send notification to:**, choose an existing SNS topic. To set the name and email addresses for a new email subscription list, choose **New list**. CloudWatch saves the list and displays it in the field so you can use it to set future alarms.

    Note
    If you use **New list** to create a new Amazon SNS topic, the email addresses must be verified before the intended recipients receive notifications. Amazon SNS sends email only when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, intended recipients don't receive a notification.
11. Choose **Next**. Add a name and optional description for your alarm. Choose **Next**.
12. Choose **Create Alarm**.

Amazon Fraud Detector Metrics

Amazon Fraud Detector sends the following metrics to CloudWatch. All metrics support these statistics: `Average`, `Minimum`, `Maximum`, `Sum`.

Metric	Description
`GetEventPrediction`	The number of GetEventPrediction API requests. Valid Dimensions: `DetectorID`
`GetEventPredictionLatency`	The interval of time taken to respond to a client request from the GetEventPrediction request. Valid Dimensions: `DetectorID` Unit: Milliseconds
`GetEventPrediction4XXError`	The number of GetEventPrediction requests where Amazon Fraud Detector returned a 4xx HTTP response code. For each 4xx response, 1 is sent. Valid Dimensions: `DetectorID`
`GetEventPrediction5XXError`	The number of GetEventPrediction requests where Amazon Fraud Detector returned a 5xx HTTP response code. For each 5xx response, 1 is sent. Valid Dimensions: `DetectorID`
`Prediction`	The number of predictions. 1 is sent if successful. Valid Dimensions: `DetectorID`, `DetectorVersionID`
`PredictionLatency`	The interval of time taken for a prediction operation. Valid Dimensions: `DetectorID`, `DetectorVersionID` Unit: Milliseconds
`PredictionError`	The number of predictions where Amazon Fraud Detector encountered an error. 1 is sent if an error is encountered.

Metric	Description
	Valid Dimensions: `DetectorID`, `DetectorVersionID`
`VariableUsed`	The number of GetEventPrediction requests where the variable was used as part of the evaluation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `VariableName`
`VariableDefaultReturned`	The number of GetEventPrediction requests where the variable was not present as part of the Event Attributes and therefore the default value for the variable was used during evaluation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `VariableName`
`RuleNotEvaluated`	The number of GetEventPrediction requests where the rule was not evaluated because a prior rule matched. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `RuleID`
`RuleEvaluateTrue`	The number of GetEventPrediction requests where the rule triggered as True and the rule outcome was returned. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `RuleID`
`RuleEvaluateFalse`	The number of GetEventPrediction requests where the rule evaluated to False. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `RuleID`
`RuleEvaluateError`	The number of GetEventPrediction requests where the rule evaluates in error Valid Dimensions: `DetectorID`, `DetectorVersionID`, `RuleID`
`OutcomeReturned`	The number of GetEventPrediction calls where the specified outcome was returned. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `OutcomeName`
`ModelInvocation (Amazon SageMaker model endpoint)`	The number of GetEventPrediction requests where the SageMaker model endpoint was invoked as part of the evaluation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `ModelEndpoint`

Metric	Description
`ModelInvocationError (Amazon SageMaker model endpoint)`	The number of GetEventPrediction requests where the invoked SageMaker model endpoint returned an error during evaluation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `ModelEndpoint`
`ModelInvocationLatency (Amazon SageMaker model endpoint)`	The interval of time taken by the Imported Model to respond as viewed from Amazon Fraud Detector. This interval includes only the model invocation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `ModelEndpoint` Unit: Milliseconds
`ModelInvocation`	The number of GetEventPrediction requests where the model was invoked as part of the evaluation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `ModelType`, `ModelID`
`ModelInvocationError`	The number of GetEventPrediction requests where the Amazon Fraud Detector model returned an error during evaluation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `ModelType`, `ModelID`
`ModelInvocationLatency`	The interval of time taken by the Amazon Fraud Detector Model to respond as viewed from Amazon Fraud Detector. This interval includes only the model invocation. Valid Dimensions: `DetectorID`, `DetectorVersionID`, `ModelType`, `ModelID` Unit: Milliseconds

Logging Amazon Fraud Detector API Calls with AWS CloudTrail

Amazon Fraud Detector is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Fraud Detector. CloudTrail captures all API calls for Amazon Fraud Detector as events, including calls from the Amazon Fraud Detector console and calls from code to the Amazon Fraud Detector APIs.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Fraud Detector. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Fraud Detector, the IP address that the request was made from, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Amazon Fraud Detector Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon Fraud Detector, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon Fraud Detector, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data that's collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon Fraud Detector supports logging every action (API operation) as an event in CloudTrail log files. For more information, see Actions.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding Amazon Fraud Detector Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `GetDetectors` operation.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "principal-id",
        "arn": "arn:aws:iam::user-arn",
        "accountId": "account-id",
        "accessKeyId": "access-key",
        "userName": "user-name"
    },
```

```
    "eventTime": "2019-11-22T02:18:03Z",
    "eventSource": "frauddetector.amazonaws.com",
    "eventName": "GetDetectors",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "source-ip-address",
    "userAgent": "aws-cli/1.11.16 Python/2.7.11 Darwin/15.6.0 botocore/1.4.73",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "request-id",
    "eventID": "event-id",
    "eventType": "AwsApiCall",
    "recipientAccountId": "recipient-account-id"
}
```

Quotas

The following tables outline Amazon Fraud Detector quotas by component.

Amazon Fraud Detector models

Resource	Default quota
Training data size	5 GB per model training
Number of models per account	100
Number of versions per model	1000
Number of deployed model versions	20
Number of concurrent training jobs total	3
Number of concurrent training jobs per model	1

Detectors / variables / outcomes / rules

Resource	Default quota
Number of variables per account	5000
Number of variables per model	100
Number of rules per account	5000
Number of event types per account	100
Number of entity types per account	100
Number of labels per account	100
Number of outcomes per account	5000
Number of detectors per account	100
Number of versions per detector	100
Number of models per detector	10

GetEventPrediction API

Resource	Default quota
Maximum GetEventPrediction API calls per second	200 TPS

Resource	Default quota
Maximum size of payload per GetEventPrediction API call	256 KB

Document history

update-history-change	update-history-description	update-history-date
Chapter rework	Rework of Getting Started and other sections	July 17, 2020
Initial release	Initial release	December 2, 2019