Entwicklerhandbuch

AWS IoT Events

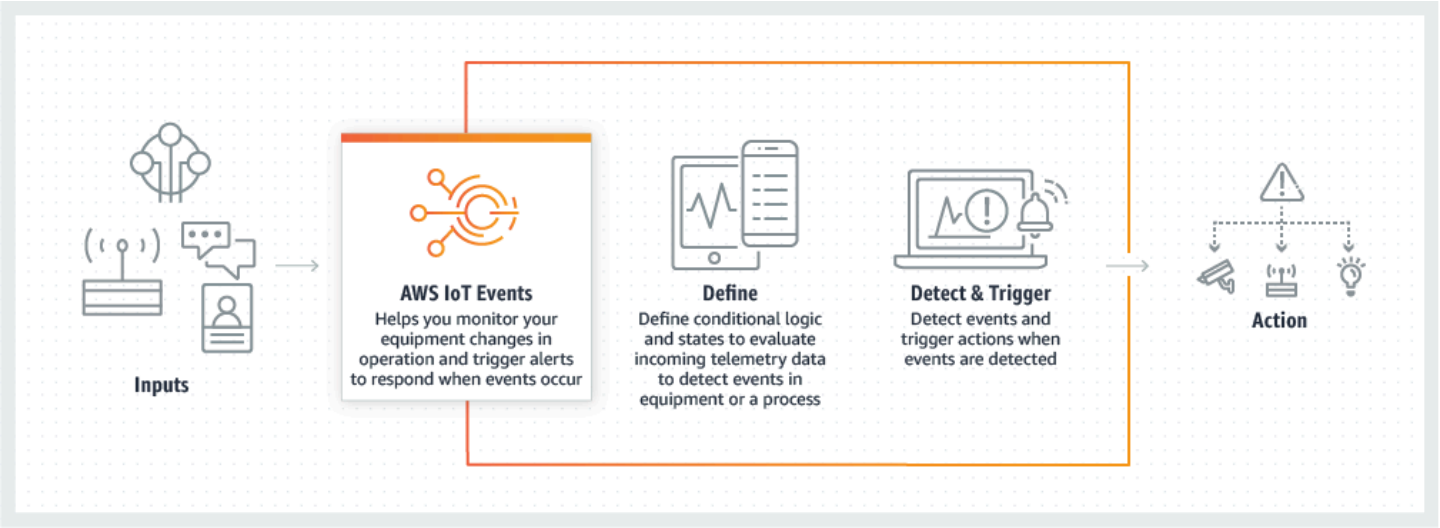
AWS IoT Events: Entwicklerhandbuch

Table of Contents

Was ist AWS IoT Events?

AWS IoT Events ermöglicht es Ihnen, Ihre Geräte oder Geräteflotten auf Ausfälle oder Betriebsänderungen zu überwachen und bei Auftreten solcher Ereignisse Maßnahmen auszulösen. AWS IoT Events überwacht kontinuierlich IoT-Sensordaten von Geräten, Prozessen, Anwendungen und anderen AWS Diensten, um wichtige Ereignisse zu identifizieren, damit Sie Maßnahmen ergreifen können.

Sie können AWS IoT Events damit komplexe Anwendungen zur Ereignisüberwachung in der AWS Cloud erstellen, auf die Sie über die AWS IoT Events Konsole oder APIs zugreifen können.



Vorteile und Funktionen

Akzeptieren Sie Eingaben aus mehreren Quellen

AWS IoT Events akzeptiert Eingaben aus vielen IoT-Telemetriedatenquellen. Dazu gehören Sensorgeräte, Verwaltungsanwendungen und andere AWS IoT Dienste wie AWS IoT Core und AWS IoT Analytics. Mithilfe einer Standard-API-Schnittstelle (`BatchPutMessage`API) können Sie alle eingegebenen Telemetriedaten AWS IoT Events per Push übertragen.

Verwenden Sie einfache logische Ausdrücke, um komplexe Ereignismuster zu erkennen

AWS IoT Events kann Muster von Ereignissen erkennen, die mehrere Eingaben von einem einzelnen IoT-Gerät oder einer einzelnen IoT-Anwendung oder von verschiedenen Geräten und vielen unabhängigen Sensoren beinhalten. Dies ist besonders nützlich, da jeder Sensor und jede Anwendung wichtige Informationen liefert. Aber nur durch die Kombination verschiedener Sensor-

und Anwendungsdaten können Sie sich ein vollständiges Bild von der Leistung und Qualität der Abläufe machen. Sie können AWS IoT Events Melder so konfigurieren, dass sie diese Ereignisse mithilfe einfacher logischer Ausdrücke anstelle von komplexem Code erkennen.

Auf Ereignissen basierende Aktionen auslösen

AWS IoT Events ermöglicht es Ihnen, Aktionen in Amazon Simple Notification Service (Amazon SNS), Lambda AWS IoT Core, Amazon SQS und Amazon Kinesis Firehose direkt auszulösen. Sie können eine AWS Lambda Funktion auch mithilfe der AWS IoT Regel-Engine auslösen, sodass Sie Aktionen mithilfe anderer Dienste wie Amazon Connect oder Ihrer eigenen ERP-Anwendungen (Enterprise Resource Planning) ausführen können.

AWS IoT Events enthält eine vorgefertigte Bibliothek mit Aktionen, die Sie ausführen können, und ermöglicht es Ihnen auch, Ihre eigenen zu definieren.

Automatische Skalierung, um den Anforderungen Ihrer Flotte gerecht zu werden

AWS IoT Events skaliert automatisch, wenn Sie homogene Geräte anschließen. Sie können einen Detektor einmal für einen bestimmten Gerätetyp definieren, und der Dienst skaliert und verwaltet automatisch alle Instanzen dieses Geräts, mit denen eine Verbindung hergestellt wird AWS IoT Events.

Anwendungsfälle

Überwachen und warten Sie Remote-Geräte

Sie müssen eine Flotte von ferngesteuerten Maschinen überwachen. Wenn eine davon nicht mehr funktioniert und Sie keinen weiteren Kontext für die Ursache des Fehlers haben, müssen Sie möglicherweise sofort die gesamte Verarbeitungseinheit oder Maschine austauschen. Aber das ist nicht nachhaltig. Mit können AWS IoT Events Sie Nachrichten von mehreren Sensoren an jedem Gerät empfangen und anhand der Fehlercodes, die im Laufe der Zeit gesendet werden, das genaue Problem diagnostizieren. Anstatt alles auszutauschen, verfügen Sie jetzt über die Informationen, die Sie benötigen, um einen Techniker zu schicken, der nur das Teil vorweisen kann, das ausgetauscht werden muss. Bei Millionen von Maschinen können sich die Einsparungen auf Millionen von Dollar summieren, wodurch Ihre Gesamtkosten für den Besitz oder die Wartung jeder Maschine sinken.

Industrieroboter verwalten

Sie setzen Roboter in Ihren Anlagen ein, um den Transport von Paketen zu automatisieren. Um die Kosten für die Roboter so gering wie möglich zu halten, verfügen die Roboter über einfache,

kostengünstige Sensoren, die Informationen an die Cloud melden. Ihre Roboter verfügen jedoch über Dutzende von Sensoren und Hunderte von Betriebsmodi, sodass es schwierig ist, Probleme zu erkennen, sobald sie auftreten. Damit können Sie ein Expertensystem aufbauen AWS IoT Events, das Sensordaten in der Cloud verarbeitet und Warnmeldungen generiert, um das technische Personal automatisch zu warnen, wenn ein Ausfall unmittelbar bevorsteht.

Verfolgen Sie Gebäudeautomationssysteme

Sie betreiben eine große Anzahl von Rechenzentren, die auf hohe Temperaturen und niedrige Luftfeuchtigkeit überwacht werden müssen, um Geräteausfälle zu verhindern, die auftreten, wenn diese Umgebungsgrenzwerte überschritten werden. Die von Ihnen verwendeten Sensoren werden von vielen Herstellern bezogen, und jeder Typ wird mit einer eigenen Verwaltungssoftware geliefert. Die Verwaltungssoftware verschiedener Anbieter ist jedoch nicht kompatibel, was die Erkennung von Problemen erschwert. Mithilfe dieser Funktion können Sie Warnmeldungen einrichten AWS IoT Events, um Ihre Betriebsanalysten rechtzeitig vor einem Ausfall über Probleme mit Ihren Heiz- und Kühlsystemen zu informieren. Auf diese Weise können Sie eine ungeplante Abschaltung des Rechenzentrums verhindern, die Tausende von Dollar für den Austausch von Geräten und potenzielle Umsatzeinbußen kosten würde.

Einrichten AWS IoT Events

Wenn Sie kein haben AWS-Konto, führen Sie die folgenden Schritte aus, um eines zu erstellen.

So registrieren Sie sich für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein registrieren AWS-Konto, Root-Benutzer des AWS-Kontos wird ein erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

Einrichten von Berechtigungen für AWS IoT Events

In diesem Abschnitt werden die Rollen und Berechtigungen beschrieben, die für die Verwendung einiger Funktionen von erforderlich sind AWS IoT Events. Sie können - AWS CLI Befehle oder die AWS Identity and Access Management (IAM)-Konsole verwenden, um Rollen und zugehörige Berechtigungsrichtlinien für den Zugriff auf Ressourcen zu erstellen oder bestimmte Funktionen in auszuführen AWS IoT Events.

Das IAM-Benutzerhandbuch enthält detailliertere Informationen zur sicheren Steuerung von Berechtigungen für den Zugriff auf - AWS Ressourcen. Spezifische Informationen zu AWS IoT Eventsfinden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS IoT Events.

Informationen zur Verwendung der IAM-Konsole zum Erstellen und Verwalten von Rollen und Berechtigungen finden Sie im IAM-Tutorial: Delegieren des Zugriffs über - AWS Konten hinweg mithilfe von IAM-Rollen.

> **Note**
>
> Schlüssel können 1–128 Zeichen lang sein und Folgendes enthalten:

- Groß- oder Kleinbuchstaben a-z
- Zahlen 0-9
- Sonderzeichen -, _ oder :.

Aktionsberechtigungen

AWS IoT Events Mit können Sie Aktionen auslösen, die andere - AWS Services verwenden. Dazu müssen Sie die AWS IoT Events Berechtigung erteilen, diese Aktionen in Ihrem Namen auszuführen. Dieser Abschnitt enthält eine Liste der Aktionen und eine Beispielrichtlinie, die die Berechtigung zum Ausführen all dieser Aktionen für Ihre Ressourcen gewährt. Ändern Sie die *Region* und die *Konto-ID*-Referenzen nach Bedarf. Wenn möglich, sollten Sie auch die Platzhalter (*) ändern, um auf bestimmte Ressourcen zu verweisen, auf die zugegriffen wird. Sie können die IAM-Konsole verwenden, um die Berechtigung AWS IoT Events zum Senden einer von Ihnen definierten Amazon SNS-Warnung zu erteilen.

AWS IoT Events unterstützt die folgenden Aktionen, mit denen Sie einen Timer verwenden oder eine Variable festlegen können:

- `setTimer` , um einen Timer zu erstellen.
- `resetTimer` , um den Timer zurückzusetzen.
- `clearTimer` , um den Timer zu löschen.
- `setVariable` , um eine Variable zu erstellen.

AWS IoT Events unterstützt die folgenden Aktionen, mit denen Sie mit - AWS Services arbeiten können:

- `iotTopicPublish` , um eine Nachricht zu einem MQTT-Thema zu veröffentlichen.
- `iotEvents` , um Daten AWS IoT Events als Eingabewert an zu senden.
- `iotSiteWise` zum Senden von Daten an eine Komponenteneigenschaft in AWS IoT SiteWise
- `dynamoDB` , um Daten an eine Amazon-DynamoDB-Tabelle zu senden.
- `dynamoDBv2` , um Daten an eine Amazon-DynamoDB-Tabelle zu senden.
- `firehose` , um Daten an einen Amazon-Data-Firehose-Stream zu senden.
- `lambda` , um eine AWS Lambda -Funktion aufzurufen.

- `sns` , um Daten als Push-Benachrichtigung zu senden.
- `sqs` , um Daten an eine Amazon SQS-Warteschlange zu senden.

Example Richtlinie

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": "iot:Publish",
       "Resource": "arn:aws:iot:<region>:<account_id>:topic/*"
     },
     {
       "Effect": "Allow",
       "Action": "iotevents:BatchPutMessage",
       "Resource": "arn:aws:iotevents:<region>:<account_id>:input/*"
     },
     {
       "Effect": "Allow",
       "Action": "iotsitewise:BatchPutAssetPropertyValue",
       "Resource": "*"
     },
     {
       "Effect": "Allow",
       "Action": "dynamodb:PutItem",
       "Resource": "arn:aws:dynamodb:<region>:<account_id>:table/*"
     },
     {
       "Effect": "Allow",
       "Action": [
         "firehose:PutRecord",
         "firehose:PutRecordBatch"
       ],
       "Resource": "arn:aws:firehose:<region>:<account_id>:deliverystream/*"
     },
     {
       "Effect": "Allow",
       "Action": "lambda:InvokeFunction",
       "Resource": "arn:aws:lambda:<region>:<account_id>:function:*"
     },
     {
```

```
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:<region>:<account_id>:*"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:<region>:<account_id>:*"
    }
  ]
}
```

Sichern von Eingabedaten

Es ist wichtig zu überlegen, wer Zugriff auf Eingabedaten zur Verwendung in einem Detektormodell gewähren kann. Wenn Sie über einen Benutzer oder eine Entität verfügen, dessen Gesamtberechtigungen Sie einschränken möchten, die jedoch berechtigt ist, ein Detektormodell zu erstellen oder zu aktualisieren, müssen Sie diesem Benutzer oder dieser Entität auch die Berechtigung erteilen, das Eingabe-Routing zu aktualisieren. Das bedeutet, dass Sie nicht nur die Berechtigung für `iotevents:CreateDetectorModel` und erteilen müssen`iotevents:UpdateDetectorModel`, sondern auch die Berechtigung für `iotevents:UpdateInputRouting`.

Example

Die folgende Richtlinie fügt die Berechtigung für hinzu`iotevents:UpdateInputRouting`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "updateRoutingPolicy",
            "Effect": "Allow",
            "Action": [
                "iotevents:UpdateInputRouting"
            ],
            "Resource": "*"
        }
    ]
}
```

Sie können eine Liste der Amazon-Ressourcennamen (ARNs) anstelle des Platzhalters „*“ für „`Resource`“ angeben, um diese Berechtigung auf bestimmte Eingaben zu beschränken. Auf diese Weise können Sie den Zugriff auf die Eingabedaten einschränken, die von Detektormodellen verbraucht werden, die vom Benutzer oder der Entität erstellt oder aktualisiert wurden.

Amazon- CloudWatch Protokollierungsrollenrichtlinie

Die folgenden Richtliniendokumente enthalten die Rollen- und Vertrauensrichtlinie, die es ermöglichen AWS IoT Events , in CloudWatch Ihrem Namen Protokolle an zu senden.

Rollenrichtlinie:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DeleteLogStream"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

Vertrauensrichtlinie:

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

```
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Sie benötigen außerdem eine IAM-Berechtigungsrichtlinie, die dem Benutzer angefügt ist und es dem Benutzer ermöglicht, Rollen wie folgt zu übergeben. Weitere Informationen finden Sie unter Erteilen von Berechtigungen an einen Benutzer zum Übergeben einer Rolle an einen - AWS Service im IAM-Benutzerhandbuch.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
          "iam:GetRole",
          "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::<account-id>:role/Role_To_Pass"
    }
  ]
}
```

Sie können den folgenden Befehl verwenden, um die Ressourcenrichtlinie für CloudWatch Protokolle festzulegen. Auf diese AWS IoT Events Weise kann Protokollereignisse in CloudWatch Streams platzieren.

```
aws logs put-resource-policy --policy-name ioteventsLoggingPolicy --policy-
document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\":
 \"IoTEventsToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service\":
 [ \"iotevents.amazonaws.com\" ] }, \"Action\":\"logs:PutLogEvents\", \"Resource\": \"*
\" } ] }"
```

Verwenden Sie den folgenden Befehl, um Protokollierungsoptionen festzulegen. Ersetzen Sie durch `roleArn` die von Ihnen erstellte Protokollierungsrolle.

```
aws iotevents put-logging-options --cli-input-json "{ \"loggingOptions\": {\"roleArn\":
 \"arn:aws:iam::123456789012:role/testLoggingRole\", \"level\": \"INFO\", \"enabled\":
 true } }"
```

Amazon SNS Messaging-Rollenrichtlinie

Die folgenden Richtliniendokumente enthalten die Rollenrichtlinie und Vertrauensrichtlinie, die das Senden von SNS-Nachrichten ermöglichen AWS IoT Events .

Rollenrichtlinie:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:sns:us-east-1:123456789012:testAction"
        }
    ]
}
```

Vertrauensrichtlinie:

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

```
    }
  ]
}
```

Erste Schritte mit der AWS IoT Events Konsole

In diesem Abschnitt erfahren Sie, wie Sie mit der AWS IoT Events Konsole ein Eingabe- und ein Detektormodell erstellen. Sie modellieren zwei Zustände eines Motors: einen Normalzustand und einen Überdruckzustand. Wenn der gemessene Druck im Motor einen bestimmten Schwellenwert überschreitet, geht das Modell vom Normalzustand in den Überdruckzustand über. Dann sendet es eine Amazon SNS SNS-Nachricht, um einen Techniker über den Zustand zu informieren. Wenn der Druck bei drei aufeinanderfolgenden Druckmessungen wieder unter den Schwellenwert fällt, kehrt das Modell in den Normalzustand zurück und sendet eine weitere Amazon SNS SNS-Nachricht als Bestätigung.

Wir prüfen, ob drei aufeinanderfolgende Messwerte unter dem Druckschwellenwert liegen, um ein mögliches Stottern bei Überdruck oder normalen Meldungen im Falle einer nichtlinearen Erholungsphase oder einer anomalen Druckmessung zu vermeiden.

Auf der Konsole finden Sie auch mehrere vorgefertigte Modellvorlagen für Melder, die Sie anpassen können. Sie können die Konsole auch verwenden, um Meldermodelle zu importieren, die von anderen geschrieben wurden, oder um Ihre Meldermodelle zu exportieren und sie in verschiedenen AWS Regionen zu verwenden. Wenn Sie ein Meldermodell importieren, stellen Sie sicher, dass Sie die erforderlichen Eingaben erstellen oder sie für die neue Region neu erstellen, und aktualisieren Sie alle verwendeten Rollen-ARNs.

Auf der Konsole finden Sie auch mehrere vorgefertigte Vorlagen für Meldermodelle, die Sie anpassen können. Sie können die Konsole auch verwenden, um Meldermodelle zu importieren, die von anderen geschrieben wurden, oder um Ihre Meldermodelle zu exportieren und sie für andere AWS-Regionen Zwecke zu verwenden. Wenn Sie ein Meldermodell importieren, stellen Sie sicher, dass Sie die erforderlichen Eingaben erstellen oder sie für die neue Region neu erstellen, und aktualisieren Sie alle verwendeten Rollen-ARNs.

Verwenden Sie die AWS IoT Events Konsole, um mehr über Folgendes zu erfahren.

Definieren Sie Eingaben

Um Ihre Geräte und Prozesse zu überwachen, müssen sie über eine Möglichkeit verfügen, Telemetriedaten in AWS IoT Events zu übertragen. Dies geschieht, indem Nachrichten als Eingaben an gesendet AWS IoT Events werden. Hierfür gibt es mehrere Möglichkeiten:

- Benutze die BatchPutMessageOperation.

- Schreiben Sie in AWS IoT Core eine AWS IoT Events Aktionsregel für die AWS IoT Regel-Engine, die Ihre Nachrichtendaten weiterleitet. AWS IoT Events Sie müssen die Eingabe anhand des Namens identifizieren.
- Verwenden Sie in AWS IoT Analytics die CreateDatasetOperation, um einen Datensatz mit zu erstellen`contentDeliveryRules`. Diese Regeln spezifizieren die AWS IoT Events Eingabe, an die der Inhalt des Datensatzes automatisch gesendet wird.

Bevor Ihre Geräte Daten auf diese Weise senden können, müssen Sie einen oder mehrere Eingänge definieren. Geben Sie dazu jeder Eingabe einen Namen und geben Sie an, welche Felder in den eingehenden Nachrichtendaten von der Eingabe überwacht werden.

Erstellen Sie ein Detektormodell

Definieren Sie mithilfe von Zuständen ein Detektormodell (ein Modell Ihrer Ausrüstung oder Ihres Prozesses). Definieren Sie für jeden Status eine bedingte (boolesche) Logik, die die eingehenden Eingaben auswertet, um signifikante Ereignisse zu erkennen. Wenn das Detektormodell ein Ereignis erkennt, kann es den Status ändern oder mithilfe anderer Dienste benutzerdefinierte oder vordefinierte Aktionen auslösen. AWS Sie können zusätzliche Ereignisse definieren, die Aktionen auslösen, wenn Sie einen Status betreten oder verlassen und optional, wenn eine Bedingung erfüllt ist.

In diesem Tutorial senden Sie eine Amazon SNS SNS-Nachricht als Aktion, wenn das Modell in einen bestimmten Status eintritt oder diesen verlässt.

Überwachen Sie ein Gerät oder einen Prozess

Wenn Sie mehrere Geräte oder Prozesse überwachen, geben Sie in jeder Eingabe ein Feld an, das das jeweilige Gerät oder den Prozess identifiziert, von dem die Eingabe stammt. Das `key` Feld finden Sie in`CreateDetectorModel`. Wenn das von identifizierte Eingabefeld einen neuen Wert `key` erkennt, wird ein neues Gerät identifiziert und ein Detektor erstellt. Jeder Detektor ist eine Instanz des Detektormodells. Der neue Detektor reagiert weiterhin auf Eingaben von diesem Gerät, bis sein Detektormodell aktualisiert oder gelöscht wird.

Wenn Sie einen einzelnen Prozess überwachen (auch wenn mehrere Geräte oder Unterprozesse Eingaben senden), geben Sie kein eindeutiges `key` Identifikationsfeld an. In diesem Fall erzeugt das Modell einen einzelnen Detektor (Instanz), wenn die erste Eingabe eingeht.

Senden Sie Nachrichten als Eingaben an Ihr Detektormodell

Es gibt mehrere Möglichkeiten, eine Nachricht von einem Gerät oder Prozess als Eingabe in einen AWS IoT Events Detektor zu senden, ohne dass Sie die Nachricht zusätzlich formatieren müssen.

In diesem Tutorial verwenden Sie die AWS IoT Konsole, um eine AWS IoT Events Aktionsregel für die AWS IoT Regel-Engine zu schreiben, in AWS IoT Events die Ihre Nachrichtendaten weitergeleitet werden.

Identifizieren Sie dazu die Eingabe anhand des Namens und verwenden Sie weiterhin die AWS IoT Konsole, um Nachrichten zu generieren, an die sie als Eingaben weitergeleitet AWS IoT Events werden.

> **Note**
>
> In diesem Tutorial wird die Konsole verwendet, um dasselbe `input` zu erstellen. Dies `detector model` wird im Beispiel unter gezeigtTutorials. Sie können dieses JSON-Beispiel verwenden, um dem Tutorial zu folgen.

Themen

Voraussetzungen

Wenn Sie noch kein AWS Konto haben, erstellen Sie eines.

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für einen anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird ein erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

3. Erstellen Sie zwei Amazon Simple Notification Service (Amazon SNS) -Themen.

   In diesem Tutorial (und dem entsprechenden Beispiel) wird davon ausgegangen, dass Sie zwei Amazon SNS SNS-Themen erstellt haben. Die ARNs dieser Themen werden wie folgt angezeigt: `arn:aws:sns:us-east-1:123456789012:underPressureAction` und. `arn:aws:sns:us-east-1:123456789012:pressureClearedAction` Ersetzen Sie diese Werte durch die ARNs der Amazon SNS SNS-Themen, die Sie erstellen. Weitere Informationen finden Sie im Amazon Simple Notification Service-Entwicklerhandbuch.

   Als Alternative zur Veröffentlichung von Benachrichtigungen zu Amazon SNS SNS-Themen können Sie die Detektoren MQTT-Nachrichten mit einem von Ihnen angegebenen Thema senden lassen. Mit dieser Option können Sie überprüfen, ob Ihr Detektormodell Instances erstellt und ob diese Instances Alerts senden, indem Sie die AWS IoT Core-Konsole verwenden, um Nachrichten zu abonnieren und zu überwachen, die zu diesen MQTT-Themen gesendet werden. Sie können den Namen des MQTT-Themas auch dynamisch zur Laufzeit definieren, indem Sie eine Eingabe oder Variable verwenden, die im Detektormodell erstellt wurde.

4. Wählen Sie einen AWS-Region , der unterstützt AWS IoT Events. Weitere Informationen finden Sie unter AWS IoT Events im Allgemeine AWS-Referenz. Hilfe finden Sie unter Arbeiten mit dem AWS Management Console im Abschnitt Erste Schritte mit dem AWS Management Console.

Erstellen Sie eine Eingabe

Wir empfehlen, bei der Erstellung der Eingaben für Ihre Modelle Dateien zusammenzustellen, die Beispielnachrichten enthalten, die Ihre Geräte oder Prozesse zur Meldung ihres Zustands senden. Diese Dateien helfen Ihnen dabei, die erforderlichen Eingaben zu definieren.

Sie können eine Eingabe mit mehreren Methoden erstellen, die in diesem Abschnitt beschrieben werden.

Erstellen Sie zunächst eine Datei mit `input.json` dem Namen Ihres lokalen Dateisystems mit dem folgenden Inhalt:

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
```

```
}
```

Nachdem Sie diese `input.json` Starterdatei haben, können Sie eine Eingabe erstellen. In einem der Themen in diesem Abschnitt finden Sie Anweisungen zum Erstellen einer Eingabe mithilfe des Navigationsbereichs oder mithilfe des Detektormodells.

Themen

- Eine Eingabe im Navigationsbereich erstellen
- Erstellen Sie eine Eingabe im Detektormodell

Eine Eingabe im Navigationsbereich erstellen

In diesem Thema wird gezeigt, wie Sie über den Navigationsbereich eine Eingabe für ein Alarm- oder ein Meldermodell erstellen.

1. Melden Sie sich bei der AWS IoT Events Konsole an oder wählen Sie die Option Neues AWS IoT Events Konto erstellen.
2. Wählen Sie in der AWS IoT Events Konsole in der oberen linken Ecke den Navigationsbereich aus und erweitern Sie ihn.
3. Wählen Sie im linken Navigationsbereich Eingaben aus.
4. Wählen Sie in der rechten Ecke der Konsole die Option Eingabe erstellen aus.
5. Geben Sie für die Eingabe eine InputNameoptionale Beschreibung ein und wählen Sie Datei hochladen. Wählen Sie im daraufhin angezeigten Dialogfeld die `input.json` Datei aus, die Sie in der Übersicht erstellt haben, um eine Eingabe zu erstellen.
6. Wählen Sie unter Eingabeattribute auswählen die zu verwendenden Attribute aus und wählen Sie Erstellen. In diesem Beispiel wählen wir motorid und sensorData.Pressure aus.

Erstellen Sie eine Eingabe im Detektormodell

In diesem Thema wird gezeigt, wie Sie einen Eingang für ein Detektormodell definieren, um Telemetriedaten oder Nachrichten zu empfangen.

1. Öffnen Sie die AWS IoT Events -Konsole.
2. Wählen Sie in der AWS IoT Events Konsole die Option Meldermodell erstellen aus.
3. Wählen Sie Neu erstellen.

4. Wählen Sie Create input (Eingabe erstellen).
5. Geben Sie für die Eingabe eine InputNameoptionale Beschreibung ein und wählen Sie Datei hochladen. Wählen Sie im daraufhin angezeigten Dialogfeld die `input.json` Datei aus, die Sie in der Übersicht erstellt haben, um eine Eingabe zu erstellen.
6. Wählen Sie unter Eingabeattribute auswählen die zu verwendenden Attribute aus und wählen Sie Erstellen. In diesem Beispiel wählen wir motorid und sensorData.Pressure aus.

Erstellen Sie ein Detektormodell

In diesem Thema definieren Sie mithilfe von Zuständen ein Detektormodell (ein Modell Ihrer Ausrüstung oder Ihres Prozesses).

Für jeden Status definieren Sie eine bedingte (boolesche) Logik, die die eingehenden Eingaben auswertet, um ein signifikantes Ereignis zu erkennen. Wenn ein Ereignis erkannt wird, ändert es den Status und kann zusätzliche Aktionen einleiten. Diese Ereignisse werden als Übergangsereignisse bezeichnet.

In Ihren Zuständen definieren Sie auch Ereignisse, die Aktionen ausführen können, wenn der Melder in diesen Zustand eintritt oder ihn verlässt oder wenn eine Eingabe eingeht (diese Ereignisse werden als `OnEnter` `OnInput` AND-Ereignisse bezeichnet). `OnExit` Die Aktionen werden nur ausgeführt, wenn die Bedingungslogik des Ereignisses Folgendes ergibt. `true`

Um ein Detektormodell zu erstellen

1. Der erste Detektorstatus wurde für Sie erstellt. Um ihn zu ändern, wählen Sie den Kreis mit der Bezeichnung State_1 im Hauptbearbeitungsbereich aus.
2. Geben Sie im Statusbereich den Namen des Bundesstaates ein und wählen Sie OnEnterEreignis hinzufügen aus.
3. Geben Sie auf der Seite „ OnEnter Ereignis hinzufügen“ einen Namen für das Ereignis und die Bedingung für das Ereignis ein. Geben Sie in diesem Beispiel ein, `true` um anzugeben, dass das Ereignis immer ausgelöst wird, wenn der Status eingegeben wird.
4. Wählen Sie unter Ereignisaktionen die Option Aktion hinzufügen aus.
5. Gehen Sie unter Ereignisaktionen wie folgt vor:

    a. Wählen Sie „Variable festlegen“

    b. Wählen Sie für den Variablenbetrieb die Option Wert zuweisen aus.

c. Geben Sie unter Variablenname den Namen der Variablen ein, die festgelegt werden soll.

d. Geben Sie für Variablenwert den Wert **`0`** (Null) ein.

6. Wählen Sie Speichern.

   Eine Variable, wie die, die Sie definiert haben, kann auf jeden Fall im Detektormodell festgelegt werden (mit einem Wert). Auf den Wert der Variablen kann erst verwiesen werden (z. B. in der bedingten Logik eines Ereignisses), wenn der Detektor einen Status erreicht hat und eine Aktion ausgeführt hat, in der er definiert oder gesetzt ist.

7. Wählen Sie im Statusbereich das X neben Status, um zur Modellpalette Detector zurückzukehren.

8. Um einen zweiten Detektorstatus zu erstellen, wählen Sie in der Modellpalette Detektor die Option Status und ziehen Sie ihn in den Hauptbearbeitungsbereich. Dadurch wird ein Zustand mit dem Titel erstelltuntitled_state_1.

9. Halten Sie im ersten Status (Normal) an. Am Rand des Bundesstaats erscheint ein Pfeil.

10. Klicken Sie auf den Pfeil und ziehen Sie ihn vom ersten Status in den zweiten Status. Eine gerichtete Linie vom ersten zum zweiten Status (mit der Bezeichnung Unbenannt) wird angezeigt.

11. Wählen Sie die Linie Ohne Titel aus. Geben Sie im Bereich „Übergangsereignis“ einen Namen für das Ereignis und eine Logik für das Ereignis ein.

12. Wählen Sie im Bereich Übergangsereignis die Option Aktion hinzufügen aus.

13. Wählen Sie im Bereich „Aktionen für Übergangsereignis hinzufügen“ die Option Aktion hinzufügen aus.

14. Wählen Sie für Aktion auswählen die Option Variable festlegen aus.

    a. Wählen Sie für Variablenoperation die Option Wert zuweisen aus.

    b. Geben Sie unter Variablenname den Namen der Variablen ein.

    c. Geben Sie für Wert zuweisen einen Wert ein, z. B.:
    `$variable.pressureThresholdBreached + 3`

    d. Wählen Sie Speichern.

15. Wählen Sie den zweiten Status untitled_state_1 aus.

16. Geben Sie im Bereich Status den Namen des Bundesstaates ein und wählen Sie für Bei Eingabe die Option Ereignis hinzufügen aus.

17. Geben Sie auf der Seite „ OnEnter Ereignis hinzufügen“ den Namen des Ereignisses und die Bedingung für das Ereignis ein. Wählen Sie Aktion hinzufügen aus.

18. Wählen Sie für Aktion auswählen die Option SNS-Nachricht senden aus.

    a. Geben Sie unter SNS-Thema den Ziel-ARN Ihres Amazon SNS SNS-Themas ein.

    b. Wählen Sie Speichern.

19. Fahren Sie mit dem Hinzufügen der Ereignisse im Beispiel fort.

    a. Wählen Sie für OnInputEreignis hinzufügen und geben Sie die folgenden Ereignisinformationen ein und speichern Sie sie.

```
Event name: Overpressurized
Event condition: $input.PressureInput.sensorData.pressure > 70
Event actions:
  Set variable:
    Variable operation: Assign value
    Variable name: pressureThresholdBreached
    Assign value: 3
```

    b. Wählen Sie für OnInput„Ereignis hinzufügen“ und geben Sie die folgenden Ereignisinformationen ein und speichern Sie sie.

```
Event name: Pressure Okay
Event condition: $input.PressureInput.sensorData.pressure <= 70
Event actions:
  Set variable:
    Variable operation: Decrement
    Variable name: pressureThresholdBreached
```

    c. Wählen Sie für OnExitEreignis hinzufügen und geben Sie die folgenden Ereignisinformationen ein und speichern Sie sie unter Verwendung des ARN des Amazon SNS SNS-Themas, das Sie erstellt haben.

```
Event name: Normal Pressure Restored
Event condition: true
Event actions:
  Send SNS message:
    Target arn: arn:aws:sns:us-east-1:123456789012:pressureClearedAction
```

20. Halten Sie im zweiten Status (Gefährlich) an. Am Rand des Bundesstaats erscheint ein Pfeil

21. Klicken Sie auf den Pfeil und ziehen Sie ihn vom zweiten Status in den ersten Status. Eine gerichtete Linie mit der Bezeichnung Unbenannt wird angezeigt.
22. Wählen Sie die Zeile „Unbenannt“ und geben Sie im Bereich „Übergangsereignis“ anhand der folgenden Informationen einen Namen für das Ereignis und eine Logik für das Ereignis ein.

    ```
    {
      Event name: BackToNormal
      Event trigger logic: $input.PressureInput.sensorData.pressure <= 70 &&
     $variable.pressureThresholdBreached <= 0
    }
    ```

    Weitere Informationen darüber, warum wir den `$input` Wert und den `$variable` Wert in der Triggerlogik testen, finden Sie im Eintrag zur Verfügbarkeit von Variablenwerten unter. Einschränkungen und Einschränkungen des Detektormodells
23. Wählen Sie den Startstatus aus. Standardmäßig wurde dieser Status erstellt, als Sie ein Detektormodell erstellt haben). Wählen Sie im Startbereich den Zielstatus aus (z. B. Normal).
24. Als Nächstes konfigurieren Sie Ihr Meldermodell so, dass es auf Eingaben wartet. Wählen Sie in der oberen rechten Ecke Veröffentlichen.
25. Gehen Sie auf der Seite Meldermodell veröffentlichen wie folgt vor.

    a. Geben Sie einen Modellnamen für den Detektor, eine Beschreibung und den Namen einer Rolle ein. Diese Rolle wurde für Sie erstellt.

    b. Wählen Sie für jeden eindeutigen Schlüsselwert einen Detektor erstellen aus. Um Ihre eigene Rolle zu erstellen und zu verwenden, folgen Sie den Schritten unter Einrichten von Berechtigungen für AWS IoT Events und geben Sie sie hier als Rolle ein.
26. Wählen Sie unter Detector creation key den Namen eines der Attribute der Eingabe, die Sie zuvor definiert haben. Das Attribut, das Sie als Schlüssel für die Erstellung des Melders wählen, muss in jeder Nachrichteneingabe vorhanden sein und für jedes Gerät, das Nachrichten sendet, eindeutig sein. In diesem Beispiel wird das motorid-Attribut verwendet.
27. Wählen Sie Save and publish (Speichern und veröffentlichen).

> **Note**
>
> Die Anzahl der eindeutigen Melder, die für ein bestimmtes Detektormodell erstellt wurden, basiert auf den gesendeten Eingangsmeldungen. Wenn ein Detektormodell erstellt wird, wird ein Schlüssel aus den Eingabeattributen ausgewählt. Dieser Schlüssel bestimmt,

welche Detektorinstanz verwendet werden soll. Wenn der Schlüssel noch nie gesehen wurde (für dieses Detektormodell), wird eine neue Detektorinstanz erstellt. Wenn der Schlüssel schon einmal gesehen wurde, verwenden wir die bestehende Detektorinstanz, die diesem Schlüsselwert entspricht.

Sie können eine Sicherungskopie Ihrer Detektormodelldefinition (in JSON) erstellen, das Detektormodell neu erstellen oder aktualisieren oder als Vorlage verwenden, um ein anderes Detektormodell zu erstellen.

Sie können dies von der Konsole aus oder mit dem folgenden CLI-Befehl tun. Ändern Sie gegebenenfalls den Namen des Detektormodells so, dass er dem entspricht, den Sie bei der Veröffentlichung im vorherigen Schritt verwendet haben.

```
aws iotevents describe-detector-model  --detector-model-name motorDetectorModel >
 motorDetectorModel.json
```

Dadurch wird eine Datei (`motorDetectorModel.json`) erstellt, deren Inhalt dem folgenden ähnelt.

```
{
    "detectorModel": {
        "detectorModelConfiguration": {
            "status": "ACTIVE",
            "lastUpdateTime": 1552072424.212,
            "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
            "creationTime": 1552072424.212,
            "detectorModelArn": "arn:aws:iotevents:us-
west-2:123456789012:detectorModel/motorDetectorModel",
            "key": "motorid",
            "detectorModelName": "motorDetectorModel",
            "detectorModelVersion": "1"
        },
        "detectorModelDefinition": {
            "states": [
                {
                    "onInput": {
                        "transitionEvents": [
                            {
                                "eventName": "Overpressurized",
                                "actions": [
                                    {
```

```
                                    "setVariable": {
                                        "variableName":
"pressureThresholdBreached",
                                        "value":
"$variable.pressureThresholdBreached + 3"
                                    }
                                }
                            ],
                            "condition": "$input.PressureInput.sensorData.pressure
> 70",
                            "nextState": "Dangerous"
                        }
                    ],
                    "events": []
                },
                "stateName": "Normal",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "init",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName":
"pressureThresholdBreached",
                                        "value": "0"
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "Back to Normal",
                            "actions": [],
```

```
                                "condition": "$variable.pressureThresholdBreached <= 1
&& $input.PressureInput.sensorData.pressure <= 70",
                                "nextState": "Normal"
                            }
                        ],
                        "events": [
                            {
                                "eventName": "Overpressurized",
                                "actions": [
                                    {
                                        "setVariable": {
                                            "variableName":
"pressureThresholdBreached",
                                            "value": "3"
                                        }
                                    }
                                ],
                                "condition": "$input.PressureInput.sensorData.pressure
> 70"
                            },
                            {
                                "eventName": "Pressure Okay",
                                "actions": [
                                    {
                                        "setVariable": {
                                            "variableName":
"pressureThresholdBreached",
                                            "value":
"$variable.pressureThresholdBreached - 1"
                                        }
                                    }
                                ],
                                "condition": "$input.PressureInput.sensorData.pressure
<= 70"
                            }
                        ]
                    },
                    "stateName": "Dangerous",
                    "onEnter": {
                        "events": [
                            {
                                "eventName": "Pressure Threshold Breached",
                                "actions": [
                                    {
```

```
                                        "sns": {
                                            "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyIoTButtonSNSTopic"
                                        }
                                    }
                                ],
                                "condition": "$variable.pressureThresholdBreached > 1"
                            }
                        ]
                    },
                    "onExit": {
                        "events": [
                            {
                                "eventName": "Normal Pressure Restored",
                                "actions": [
                                    {
                                        "sns": {
                                            "targetArn": "arn:aws:sns:us-
west-2:123456789012:IoTVirtualButtonTopic"
                                        }
                                    }
                                ],
                                "condition": "true"
                            }
                        ]
                    }
                }
            ],
            "initialStateName": "Normal"
        }
    }
}
```

Senden Sie Eingaben, um das Detektormodell zu testen

Es gibt mehrere Möglichkeiten, Telemetriedaten zu empfangen AWS IoT Events (siehe Unterstützte Aktionen). In diesem Thema erfahren Sie, wie Sie in der AWS IoT Konsole eine AWS IoT Regel erstellen, die Nachrichten als Eingaben an Ihren AWS IoT Events Detektor weiterleitet. Sie können den MQTT-Client der AWS IoT Konsole verwenden, um Testnachrichten zu senden. Mit dieser Methode können Sie Telemetriedaten darüber abrufen, AWS IoT Events wann Ihre Geräte MQTT-Nachrichten über den AWS IoT Message Broker senden können.

Um Eingaben zu senden, um das Detektormodell zu testen

1. Öffnen Sie die AWS IoT Core -Konsole. Wählen Sie im linken Navigationsbereich unter Verwalten die Option Nachrichtenweiterleitung und anschließend Regeln aus.
2. Wählen Sie oben rechts Regel erstellen aus.
3. Führen Sie auf der Seite Regel erstellen die folgenden Schritte aus:

   1. Schritt 1. Geben Sie die Eigenschaften der Regel an. Füllen Sie die folgenden Felder aus:
      - Name der Regel. Geben Sie einen Namen für Ihre Regel ein, z. `MyIoTEventsRule` B.

        > **Note**
        >
        > Verwenden Sie keine Leerzeichen.

      - Beschreibung der Regel. Dieser Schritt ist optional.
      - Wählen Sie Weiter aus.
   2. Schritt 2. Konfigurieren Sie die SQL-Anweisung. Füllen Sie die folgenden Felder aus:
      - SQL-Version. Wählen Sie die entsprechende Option aus der Liste aus.
      - SQL-Anweisung. Geben Sie **`SELECT *, topic(2) as motorid FROM 'motors/+/status'`** ein.

      Wählen Sie Weiter aus.
   3. Schritt 3. Regelaktionen anhängen. Gehen Sie im Abschnitt Regelaktionen wie folgt vor:
      - Aktion 1. Wählen Sie IoT Events aus. Die folgenden Felder werden angezeigt:
        a. Geben Sie den Namen ein. Wählen Sie die entsprechende Option aus der Liste aus. Wenn Ihre Eingabe nicht angezeigt wird, wählen Sie Aktualisieren.

        Um einen neuen Input zu erstellen, wählen Sie Create IoT Events input. Füllen Sie die folgenden Felder aus:

        - Geben Sie den Namen ein. Geben Sie `PressureInput` ein.
        - Beschreibung. Dieser Schritt ist optional.
        - Laden Sie eine JSON-Datei hoch. Laden Sie eine Kopie Ihrer JSON-Datei hoch. Auf diesem Bildschirm befindet sich ein Link zu einer Beispieldatei, falls Sie keine Datei haben. Der Code beinhaltet:

          ```
          {
          ```

```
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

- Wählen Sie Eingabeattribute. Wählen Sie die entsprechende (n) Option (en) aus.
- Tags. Dieser Schritt ist optional.

Wählen Sie Erstellen.

Kehren Sie zum Bildschirm Regel erstellen zurück und aktualisieren Sie das Feld Eingabename. Wählen Sie die Eingabe aus, die Sie gerade erstellt haben.

b. Batch-Modus. Dieser Schritt ist optional. Wenn es sich bei der Payload um eine Reihe von Nachrichten handelt, wählen Sie diese Option.

c. Nachrichten-ID. Dies ist zwar optional, wird aber empfohlen.

d. IAM role (IAM-Rolle. Wählen Sie die entsprechende Rolle aus der Liste aus. Wenn die Rolle nicht aufgeführt ist, wählen Sie Neue Rolle erstellen aus.

Geben Sie einen Rollennamen ein und wählen Sie Erstellen aus.

Um eine weitere Regel hinzuzufügen, wählen Sie Regelaktion hinzufügen

- Fehleraktion. Dieser Abschnitt ist optional. Um eine Aktion hinzuzufügen, wählen Sie Fehleraktion hinzufügen und wählen Sie die entsprechende Aktion aus der Liste aus.

Füllen Sie die angezeigten Felder aus.

- Wählen Sie Weiter aus.

4. Schritt 4. Überprüfen und erstellen. Überprüfen Sie die Informationen auf dem Bildschirm und wählen Sie Erstellen.

4. Wählen Sie im linken Navigationsbereich unter Test die Option MQTT-Testclient aus.

5. Wählen Sie Publish to topic (In Thema veröffentlichen) aus. Füllen Sie die folgenden Felder aus:

- Name des Themas. Geben Sie einen Namen ein, um die Nachricht zu identifizieren, z. `motors/Fulton-A32/status` B.
- Nutzlast der Nachricht. Geben Sie Folgendes ein:

```
{
```

```
  "messageId": 100,
  "sensorData": {
    "pressure": 39
  }
}
```

> **Note**
>
> Ändern Sie die `messageId` jedes Mal, wenn Sie eine neue Nachricht veröffentlichen.

6. Behalten Sie für Publish das gleiche Thema bei, ändern Sie das `"pressure"` in der Payload jedoch auf einen Wert, der über dem Schwellenwert liegt, den Sie im Detektormodell angegeben haben (z. B.**85**).
7. Wählen Sie Publish.

Die von Ihnen erstellte Detector-Instance generiert und sendet Ihnen eine Amazon SNS SNS-Nachricht. Senden Sie weiterhin Nachrichten mit Druckwerten über oder unter dem Druckgrenzwert (70 in diesem Beispiel), um zu sehen, wie der Detektor in Betrieb ist.

In diesem Beispiel müssen Sie drei Nachrichten mit Druckwerten unter dem Schwellenwert senden, um in den Normalzustand zurückzukehren und eine Amazon SNS SNS-Meldung zu erhalten, die darauf hinweist, dass der Überdruckzustand behoben ist. Sobald der Melder wieder im Normalzustand ist, wechselt der Detektor durch eine Meldung mit einem Druckwert über dem Grenzwert in den Status Gefährlich und sendet eine Amazon SNS SNS-Meldung, die diesen Zustand anzeigt.

Nachdem Sie nun ein einfaches Eingabe- und Meldermodell erstellt haben, versuchen Sie Folgendes.

- Weitere Beispiele (Vorlagen) für Detektormodelle finden Sie auf der Konsole.
- Folgen Sie den Schritten unterEinfaches step-by-step Beispiel, um ein Eingabe- und Detektormodell mit dem zu erstellen AWS CLI
- Erfahren Sie mehr über die in den Ereignissen Ausdrücke verwendeten.
- Erfahren Sie mehr über Unterstützte Aktionen.
- Wenn etwas nicht funktioniert, finden Sie weitere Informationen unterFehlerbehebung für AWS IoT Events.

Bewährte Verfahren für AWS IoT Events

Folgen Sie diesen bewährten Methoden, um den größtmöglichen Nutzen daraus zu ziehen. AWS IoT Events

Themen

Aktivieren Sie die CloudWatch Amazon-Protokollierung bei der Entwicklung von AWS IoT Events Detektormodellen

Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Damit CloudWatch erhalten Sie systemweiten Einblick in die Ressourcennutzung, die Anwendungsleistung und den Betriebszustand. Wenn Sie ein AWS IoT Events Detektormodell entwickeln oder debuggen, CloudWatch wissen Sie, was AWS IoT Events gerade passiert und welche Fehler dabei auftreten.

Um zu aktivieren CloudWatch

1. Falls Sie dies noch nicht getan haben, folgen Sie den Schritten unter Einrichten von Berechtigungen für AWS IoT Events So erstellen Sie eine Rolle mit einer angehängten Richtlinie, die die Berechtigung zum Erstellen und Verwalten von CloudWatch Protokollen für erteilt AWS IoT Events.
2. Rufen Sie die AWS IoT Events -Konsole auf.
3. Wählen Sie im Navigationsbereich Settings (Einstellungen).
4. Wählen Sie auf der Seite Einstellungen die Option Bearbeiten aus.
5. Gehen Sie auf der Seite Protokollierungsoptionen bearbeiten im Abschnitt Protokollierungsoptionen wie folgt vor:

    a. Wählen Sie unter Ausführlichkeitsgrad eine Option aus.

b. Wählen Sie unter Rolle auswählen eine Rolle aus, die über ausreichende Berechtigungen verfügt, um die von Ihnen ausgewählten Protokollierungsaktionen durchzuführen.

c. (Optional) Wenn Sie Debug als Ausführlichkeitsstufe ausgewählt haben, können Sie Debug-Ziele hinzufügen, indem Sie wie folgt vorgehen:

   i. Wählen Sie unter Debug-Ziele die Option Modelloption hinzufügen aus.

   ii. Geben Sie einen Modellnamen für den Detektor ein und (optional) KeyValue, um die Meldermodelle und spezifischen Melder (Instanzen) anzugeben, die protokolliert werden sollen.

6. Wählen Sie Aktualisieren.

Ihre Protokollierungsoptionen wurden erfolgreich aktualisiert.

Veröffentlichen Sie regelmäßig, um Ihr Meldermodell zu speichern, wenn Sie in der AWS IoT Events Konsole arbeiten

Wenn Sie die AWS IoT Events Konsole verwenden, werden Ihre laufenden Arbeiten lokal in Ihrem Browser gespeichert. Sie müssen jedoch Veröffentlichen wählen, um Ihr Meldermodell zu speichern AWS IoT Events. Nachdem Sie ein Detektormodell veröffentlicht haben, ist Ihr veröffentlichtes Werk in jedem Browser verfügbar, den Sie für den Zugriff auf Ihr Konto verwenden.

> **Note**
>
> Wenn Sie Ihr Werk nicht veröffentlichen, wird es nicht gespeichert. Nachdem Sie ein Detektormodell veröffentlicht haben, können Sie seinen Namen nicht mehr ändern. Sie können seine Definition jedoch weiter ändern.

Speichern Sie Ihre AWS IoT Events Daten, um einen möglichen Datenverlust aufgrund einer langen Zeit der Inaktivität zu vermeiden

Wenn Sie sie über einen AWS IoT Events längeren Zeitraum nicht verwenden, werden Ihre Daten, einschließlich Ihrer Meldermodelle, möglicherweise automatisch gelöscht. Ein erheblicher Zeitraum könnte beispielsweise bedeuten, dass Ihnen keine Gebühren entstehen und Sie keine

Detektormodelle erstellen. Wir werden jedoch keine Daten oder Detektormodelle löschen, ohne Sie mindestens 30 Tage vorher darüber zu informieren. Wenn Sie Daten über einen längeren Zeitraum speichern müssen, sollten Sie die Nutzung von AWS Speicherdiensten in Betracht ziehen.

Tutorials

In diesem Kapitel erfahren Sie, wie Sie:

- Holen Sie sich Hilfe bei der Entscheidung, welche Zustände in Ihr Detektormodell aufgenommen werden sollen, und bestimmen Sie, ob Sie eine oder mehrere Detektorinstanzen benötigen.
- Folgen Sie einem Beispiel, das den verwendetAWS CLI.
- Erstellen Sie eine Eingabe für den Empfang von Telemetriedaten von einem Gerät und einem Detektormodell, um den Status des Geräts, das diese Daten sendet, zu überwachen und darüber zu berichten.
- Informieren Sie sich über die Einschränkungen und Beschränkungen für Eingänge, Meldermodelle und den AWS IoT Events Service.
- Sehen Sie sich ein komplexeres Beispiel für ein Detektormodell mit Kommentaren an.

Themen

Wird AWS IoT Events zur Überwachung Ihrer IoT-Geräte verwendet

Sie können AWS IoT Events damit Ihre Geräte oder Prozesse überwachen und bei wichtigen Ereignissen Maßnahmen ergreifen. Gehen Sie dazu wie folgt vor:

Eingaben erstellen

Sie müssen über eine Möglichkeit verfügen, mit der Ihre Geräte und Prozesse Telemetriedaten abrufen können. AWS IoT Events Sie tun dies, indem Sie Nachrichten als Eingaben an AWS IoT Events senden. Sie können Nachrichten auf verschiedene Arten als Eingaben senden:

- Verwenden Sie die BatchPutMessageOperation.
- Definieren Sie eine `iotEvents`Regelaktion für die AWS IoT Core Regel-Engine. Die Regelaktion leitet Nachrichtendaten aus Ihrer Eingabe an. AWS IoT Events

- Verwenden Sie in die CreateDatasetOperationAWS IoT Analytics, um einen Datensatz mit zu erstellen. `contentDeliveryRules` Diese Regeln spezifizieren die AWS IoT Events Eingabe, an die der Inhalt des Datensatzes automatisch gesendet wird.
- Definieren Sie eine IoTEvents-Aktion in einem AWS IoT Events Detektormodell `onExit` oder `transitionEvents` einem `onInput` Ereignis. Informationen über die Detektormodellinstanz und das Ereignis, das die Aktion ausgelöst hat, werden als Eingabe mit dem von Ihnen angegebenen Namen an das System zurückgemeldet.

Bevor Ihre Geräte Daten auf diese Weise senden, müssen Sie einen oder mehrere Eingänge definieren. Geben Sie dazu jeder Eingabe einen Namen und geben Sie an, welche Felder in den eingehenden Nachrichtendaten von der Eingabe überwacht werden. AWS IoT Eventserhält seine Eingabe in Form von JSON-Nutzdaten aus vielen Quellen. Jede Eingabe kann eigenständig bearbeitet oder mit anderen Eingaben kombiniert werden, um komplexere Ereignisse zu erkennen.

Erstellen Sie ein Detektormodell

Definieren Sie mithilfe von Zuständen ein Detektormodell (ein Modell Ihrer Ausrüstung oder Ihres Prozesses). Für jeden Zustand definieren Sie eine bedingte (boolesche) Logik, die die eingehenden Eingaben auswertet, um signifikante Ereignisse zu erkennen. Wenn ein Ereignis erkannt wird, kann es den Status ändern oder mithilfe anderer AWS Dienste benutzerdefinierte oder vordefinierte Aktionen auslösen. Sie können zusätzliche Ereignisse definieren, die Aktionen auslösen, wenn Sie einen Status betreten oder verlassen und optional, wenn eine Bedingung erfüllt ist.

In diesem Tutorial senden Sie eine Amazon SNS SNS-Nachricht als Aktion, wenn das Modell in einen bestimmten Status eintritt oder diesen verlässt.

Überwachen Sie ein Gerät oder einen Prozess

Wenn Sie mehrere Geräte oder Prozesse überwachen, geben Sie in jeder Eingabe ein Feld an, das das jeweilige Gerät oder den Prozess identifiziert, von dem die Eingabe stammt. (Siehe das `key` Feld unter`CreateDetectorModel`.) Wenn ein neues Gerät identifiziert wird (ein neuer Wert wird in dem durch das identifizierten Eingabefeld angezeigt`key`), wird ein Detektor erzeugt. (Jeder Detektor ist eine Instanz des Detektormodells.) Dann reagiert der neue Detektor weiterhin auf Eingaben von diesem Gerät, bis sein Meldermodell aktualisiert oder gelöscht wird.

Wenn Sie einen einzelnen Prozess überwachen (auch wenn mehrere Geräte oder Unterprozesse Eingaben senden), geben Sie kein eindeutiges `key` Identifikationsfeld an. In diesem Fall wird ein einzelner Detektor (Instanz) erstellt, wenn die erste Eingabe eintrifft.

Senden Sie Nachrichten als Eingaben an Ihr Detektormodell

Es gibt mehrere Möglichkeiten, eine Nachricht von einem Gerät oder Prozess als Eingabe in einen AWS IoT Events Detektor zu senden, ohne dass Sie die Nachricht zusätzlich formatieren müssen. In diesem Tutorial verwenden Sie die AWS IoT Konsole, um eine AWS IoT EventsAktionsregel für die AWS IoT Core Regel-Engine zu schreiben, in AWS IoT Events die Ihre Nachrichtendaten weitergeleitet werden. Dazu identifizieren Sie die Eingabe anhand des Namens. Anschließend verwenden Sie weiterhin die AWS IoT Konsole, um einige Nachrichten zu generieren, an die Sie als Eingaben weitergeleitet werdenAWS IoT Events.

Woher wissen Sie, welche Zustände Sie in einem Detektormodell benötigen?

Um zu bestimmen, welche Zustände Ihr Detektormodell haben sollte, entscheiden Sie zunächst, welche Maßnahmen Sie ergreifen können. Wenn Ihr Auto beispielsweise mit Benzin betrieben wird, schauen Sie bei Fahrtantritt auf die Tankanzeige, ob Sie tanken müssen. Hier haben Sie eine Aktion: Sagen Sie dem Fahrer, er solle „Gas holen“. Ihr Meldermodell benötigt zwei Zustände: „Auto benötigt keinen Kraftstoff“ und „Auto benötigt Kraftstoff“. Im Allgemeinen möchten Sie einen Status für jede mögliche Aktion und einen weiteren für den Fall definieren, dass keine Aktion erforderlich ist. Das funktioniert auch dann, wenn die Aktion selbst komplizierter ist. Möglicherweise möchten Sie Informationen darüber suchen, wo sich die nächstgelegene Tankstelle oder der günstigste Preis befindet, und diese Informationen hinzufügen, aber Sie tun dies, wenn Sie die Nachricht „Geh tanken“ senden.

Um zu entscheiden, welchen Status Sie als Nächstes eingeben möchten, schauen Sie sich die Eingaben an. Die Eingaben enthalten die Informationen, die Sie benötigen, um zu entscheiden, in welchem Zustand Sie sich befinden sollten. Um eine Eingabe zu erstellen, wählen Sie eines oder mehrere Felder in einer von Ihrem Gerät oder Prozess gesendeten Nachricht aus, die Ihnen bei der Entscheidung helfen. In diesem Beispiel benötigen Sie eine Eingabe, die Ihnen den aktuellen Kraftstoffstand („Prozent voll“) anzeigt. Vielleicht sendet Ihnen Ihr Auto mehrere verschiedene Nachrichten mit jeweils unterschiedlichen Feldern. Um diese Eingabe zu erstellen, müssen Sie die Nachricht und das Feld auswählen, das den aktuellen Füllstand der Gasanzeige anzeigt. Die Länge der Reise, die Sie unternehmen werden („Entfernung zum Ziel“), kann aus Gründen der Übersichtlichkeit fest codiert werden. Sie können Ihre durchschnittliche Reisedauer verwenden. Auf der Grundlage der Eingabe führen Sie einige Berechnungen durch (wie viele Gallonen entspricht dieser volle Prozentsatz? ist die durchschnittliche Reisedauer größer als die Meilen, die Sie zurücklegen können, wenn man die Gallonen berücksichtigt, die Sie haben, und Ihren

durchschnittlichen „Meilen pro Gallone“. Sie führen diese Berechnungen durch und senden bei Ereignissen Nachrichten.

Bisher haben Sie zwei Zustände und eine Eingabe. Sie benötigen ein Ereignis im ersten Status, das die Berechnungen auf der Grundlage der Eingabe durchführt und entscheidet, ob in den zweiten Status übergegangen werden soll. Das ist ein Übergangsereignis. (`transitionEvents`befinden sich in der `onInput` Ereignisliste eines Bundesstaates. Beim Empfang einer Eingabe in diesem ersten Zustand wechselt das Ereignis in den zweiten Status, sofern der Zustand des Ereignisses erfüllt `condition` ist.) Wenn Sie den zweiten Status erreichen, senden Sie die Nachricht, sobald Sie den Status betreten. (Sie verwenden ein `onEnter` Ereignis. Beim Eintritt in den zweiten Status sendet dieses Ereignis die Nachricht. Sie müssen nicht warten, bis eine weitere Eingabe eintrifft.) Es gibt auch andere Arten von Ereignissen, aber das ist alles, was Sie für ein einfaches Beispiel benötigen.

Die anderen Arten von Ereignissen sind `onExit` und`onInput`. Sobald eine Eingabe eingeht und die Bedingung erfüllt ist, führt ein `onInput` Ereignis die angegebenen Aktionen aus. Wenn ein Vorgang seinen aktuellen Status verlässt und die Bedingung erfüllt ist, führt das `onExit` Ereignis die angegebenen Aktionen aus.

Fehlt dir etwas? Ja, wie kehrt man zum ersten Zustand „Auto braucht keinen Kraftstoff“ zurück? Nachdem Sie Ihren Benzintank gefüllt haben, zeigt der Eingang einen vollen Tank an. In Ihrem zweiten Zustand benötigen Sie ein Übergangsereignis, das zum ersten Zustand zurückkehrt. Dieses Ereignis tritt ein, wenn die Eingabe empfangen wird (in den `onInput:` Ereignissen des zweiten Zustands). Es sollte in den ersten Zustand zurückkehren, wenn die Berechnungen ergeben, dass Sie jetzt genug Benzin haben, um dorthin zu gelangen, wo Sie hin möchten.

Das sind die Grundlagen. Einige Detektormodelle werden komplexer, wenn sie Zustände hinzufügen, die wichtige Eingaben widerspiegeln, nicht nur mögliche Aktionen. In einem Detektormodell, das die Temperatur erfasst, könnten beispielsweise drei Zustände vorliegen: ein „normaler“ Zustand, ein „zu heißer“ Zustand und ein Zustand mit „potenziellem Problem“. Sie gehen in den potenziellen Problemzustand über, wenn die Temperatur über ein bestimmtes Niveau steigt, aber noch nicht zu heiß geworden ist. Sie möchten keinen Alarm senden, es sei denn, die Temperatur bleibt länger als 15 Minuten bei dieser Temperatur. Wenn sich die Temperatur vorher wieder normalisiert hat, geht der Melder wieder in den Normalzustand über. Wenn der Timer abläuft, wechselt der Melder in den zu heißen Zustand und sendet einen Alarm, nur um vorsichtig zu sein. Sie könnten dasselbe tun, indem Sie Variablen und einen komplexeren Satz von Ereignisbedingungen verwenden. Oft ist es jedoch einfacher, einen anderen Status zu verwenden, um die Ergebnisse Ihrer Berechnungen tatsächlich zu speichern.

Woher wissen Sie, ob Sie eine oder mehrere Instanzen eines Detektors benötigen?

Um zu entscheiden, wie viele Instanzen Sie benötigen, fragen Sie sich: „Was möchten Sie wissen?“ Nehmen wir an, Sie möchten wissen, wie das Wetter heute ist. Regnet es (Bundesstaat)? Müssen Sie einen Regenschirm mitnehmen (Aktion)? Sie können einen Sensor verwenden, der die Temperatur meldet, einen anderen, der die Luftfeuchtigkeit meldet, und andere, die den Luftdruck, die Windgeschwindigkeit und -richtung sowie den Niederschlag melden. Sie müssen jedoch alle diese Sensoren gemeinsam überwachen, um den Wetterzustand (Regen, Schnee, Bewölkung, Sonne) und die entsprechenden Maßnahmen zu ermitteln (nehmen Sie sich einen Regenschirm oder tragen Sie Sonnencreme auf). Trotz der Anzahl der Sensoren möchten Sie, dass eine Melderinstanz den Wetterstatus überwacht und Sie darüber informiert, welche Maßnahmen zu ergreifen sind.

Wenn Sie jedoch für die Wettervorhersage in Ihrer Region verantwortlich sind, verfügen Sie möglicherweise über mehrere Instanzen solcher Sensoranordnungen, die sich an verschiedenen Orten in der Region befinden. Die Menschen an jedem Standort müssen wissen, wie das Wetter an diesem Ort ist. In diesem Fall benötigen Sie mehrere Instanzen Ihres Melders. Die von jedem Sensor an jedem Standort gemeldeten Daten müssen ein Feld enthalten, das Sie als `key` Feld festgelegt haben. Dieses Feld ermöglicht es, eine Melderinstanz für den Bereich AWS IoT Events zu erstellen und diese Informationen dann weiterhin an diese Melderinstanz weiterzuleiten, sobald sie eintreffen. Keine ruinierten Haare oder sonnenverbrannten Nasen mehr!

Im Grunde benötigen Sie eine Melderinstanz, wenn Sie eine Situation (einen Prozess oder einen Standort) überwachen müssen. Wenn Sie viele Situationen (Standorte, Prozesse) überwachen müssen, benötigen Sie mehrere Melderinstanzen.

Einfaches step-by-step Beispiel

In diesem Beispiel rufen wir die AWS IoT Events APIs mithilfe von AWS CLI Befehlen auf, um einen Detektor zu erstellen, der zwei Zustände eines Motors modelliert: einen Normalzustand und einen Überdruckzustand.

Wenn der gemessene Druck im Motor einen bestimmten Schwellenwert überschreitet, wechselt das Modell in den Überdruckzustand und sendet eine Amazon Simple Notification Service (Amazon SNS) -Meldung, um einen Techniker über den Zustand zu informieren. Wenn der Druck bei drei aufeinanderfolgenden Druckmessungen unter den Schwellenwert fällt, kehrt das Modell in den Normalzustand zurück und sendet eine weitere Amazon SNS SNS-Nachricht als Bestätigung,

dass der Zustand behoben ist. Wir benötigen drei aufeinanderfolgende Messwerte unter dem Druckschwellenwert, um bei einer nichtlinearen Erholungsphase oder einem einmaligen anomalen Wiederaufnahmewert mögliche ruckelnde Meldungen zu vermeiden.

Im Folgenden finden Sie eine Übersicht über die Schritte zur Erstellung des Melders.

Erstellen Sie Eingaben.

Um Ihre Geräte und Prozesse zu überwachen, müssen sie über eine Möglichkeit verfügen, Telemetriedaten in AWS IoT Events zu übertragen. Dies geschieht durch Senden von Nachrichten als Eingaben an AWS IoT Events. Hierfür gibt es mehrere Möglichkeiten:

- Verwenden Sie die BatchPutMessageOperation. Diese Methode ist einfach, setzt jedoch voraus, dass Ihre Geräte oder Prozesse über ein SDK oder das auf die AWS IoT Events API zugreifen könnenAWS CLI.
- Schreiben Sie in AWS IoT Core eine AWS IoT EventsAktionsregel für die AWS IoT Core Regel-Engine, in AWS IoT Events die Ihre Nachrichtendaten weitergeleitet werden. Dadurch wird die Eingabe anhand des Namens identifiziert. Verwenden Sie diese Methode, wenn Ihre Geräte oder Prozesse Nachrichten senden können oder dies bereits tunAWS IoT Core. Diese Methode erfordert in der Regel weniger Rechenleistung von einem Gerät.
- Verwenden Sie die CreateDatasetOperationAWS IoT Analytics, um einen Datensatz mit `contentDeliveryRules` Angabe der AWS IoT Events Eingabe zu erstellen, wobei der Inhalt des Datensatzes automatisch gesendet wird. Verwenden Sie diese Methode, wenn Sie Ihre Geräte oder Prozesse auf der Grundlage aggregierter oder analysierter Daten steuern möchten. AWS IoT Analytics

Bevor Ihre Geräte Daten auf diese Weise senden können, müssen Sie eine oder mehrere Eingaben definieren. Geben Sie dazu jeder Eingabe einen Namen und geben Sie an, welche Felder in den eingehenden Nachrichtendaten von der Eingabe überwacht werden sollen.

Erstellen Sie ein Detektormodell

Erstellen Sie mithilfe von Zuständen ein Detektormodell (ein Modell Ihrer Ausrüstung oder Ihres Prozesses). Definieren Sie für jeden Status eine bedingte (boolesche) Logik, die die eingehenden Eingaben auswertet, um signifikante Ereignisse zu erkennen. Wenn ein Ereignis erkannt wird, kann es den Status ändern oder mithilfe anderer Dienste benutzerdefinierte oder vordefinierte Aktionen auslösen. AWS Sie können zusätzliche Ereignisse definieren, die Aktionen auslösen, wenn Sie einen Status betreten oder verlassen und optional, wenn eine Bedingung erfüllt ist.

Überwachen Sie mehrere Geräte oder Prozesse

Wenn Sie mehrere Geräte oder Prozesse überwachen und diese einzeln verfolgen möchten, geben Sie in jeder Eingabe ein Feld an, das das jeweilige Gerät oder den Prozess identifiziert, von dem die Eingabe stammt. Das `key` Feld finden Sie in`CreateDetectorModel`. Wenn ein neues Gerät identifiziert wird (ein neuer Wert wird in dem durch das identifizierten Eingabefeld angezeigt`key`), wird eine Melderinstanz erstellt. Die neue Melderinstanz reagiert weiterhin auf Eingaben von diesem bestimmten Gerät, bis ihr Meldermodell aktualisiert oder gelöscht wird. Sie haben so viele eindeutige Detektoren (Instanzen), wie es eindeutige Werte in den `key` Eingabefeldern gibt.

Überwachen Sie ein einzelnes Gerät oder einen Prozess

Wenn Sie einen einzelnen Prozess überwachen (auch wenn mehrere Geräte oder Unterprozesse Eingaben senden), geben Sie kein eindeutiges `key` Identifikationsfeld an. In diesem Fall wird ein einzelner Detektor (Instanz) erstellt, wenn die erste Eingabe eintrifft. Beispielsweise könnten Sie in jedem Raum eines Hauses Temperatursensoren haben, aber nur eine HLK-Einheit, um das gesamte Haus zu heizen oder zu kühlen. Sie können dies also nur als einen einzigen Vorgang steuern, auch wenn jeder Raumnutzer möchte, dass seine Stimme (Eingabe) Vorrang hat.

Senden Sie Nachrichten von Ihren Geräten oder Prozessen als Eingaben in Ihr Meldermodell

In den Eingängen haben wir die verschiedenen Möglichkeiten beschrieben, eine Nachricht von einem Gerät oder Prozess als Eingabe in einen AWS IoT Events Detektor zu senden. Nachdem Sie die Eingänge erstellt und das Detektormodell erstellt haben, können Sie mit dem Senden von Daten beginnen.

> **Note**
>
> Wenn Sie ein Meldermodell erstellen oder ein vorhandenes aktualisieren, dauert es einige Minuten, bis das neue oder aktualisierte Meldermodell Nachrichten empfängt und Melder (Instanzen) erstellt. Wenn das Meldermodell aktualisiert wird, kann es sein, dass Sie während dieser Zeit weiterhin ein Verhalten beobachten, das auf der vorherigen Version basiert.

Themen

- Erstellen Sie einen Eingang zur Erfassung von Gerätedaten
- Erstellen Sie ein Meldermodell zur Darstellung von Gerätezuständen

- Sendet Nachrichten als Eingaben an einen Detektor

Erstellen Sie einen Eingang zur Erfassung von Gerätedaten

Nehmen wir als Beispiel an, Ihre Geräte senden Nachrichten im folgenden Format.

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

Mit dem folgenden AWS CLI Befehl können Sie eine Eingabe erstellen, um die `pressure` Daten und die `motorid` (die das spezifische Gerät identifiziert, das die Nachricht gesendet hat) zu erfassen.

```
aws iotevents create-input  --cli-input-json file://pressureInput.json
```

Die Datei `pressureInput.json` enthält Folgendes.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.pressure" },
      { "jsonPath": "motorid" }
    ]
  }
}
```

Wenn Sie Ihre eigenen Eingaben erstellen, denken Sie daran, zunächst Beispielnachrichten als JSON-Dateien von Ihren Geräten oder Prozessen zu sammeln. Sie können sie verwenden, um eine Eingabe von der Konsole oder der CLI aus zu erstellen.

Erstellen Sie ein Meldermodell zur Darstellung von Gerätezuständen

In Erstellen Sie einen Eingang zur Erfassung von Gerätedaten haben Sie ein auf einer Nachricht `input` basierendes Gerät erstellt, das Druckdaten eines Motors meldet. Um mit dem Beispiel fortzufahren: Hier ist ein Detektormodell, das auf ein Überdruckereignis in einem Motor reagiert.

Sie erstellen zwei Zustände: "`Normal`„und"`Dangerous`“. Jeder Detektor (Instanz) geht bei seiner Erstellung in den Zustand `Normal` "" über. Die Instanz wird erstellt, wenn eine Eingabe mit einem eindeutigen Wert für `key` "`motorid`" eingeht.

Wenn die Melder-Instance einen Druckwert von 70 oder mehr empfängt, wechselt sie in den Status `Dangerous` "" und sendet eine Amazon SNS SNS-Nachricht als Warnung. Wenn die Druckwerte bei drei aufeinanderfolgenden Eingängen wieder normal sind (weniger als 70), kehrt der Detektor in den Zustand "`Normal`" zurück und sendet eine weitere Amazon SNS SNS-Meldung als Entwarnung.

Dieses Beispiel-Detektormodell geht davon aus, dass Sie zwei Amazon SNS SNS-Themen erstellt haben, deren Amazon-Ressourcennamen (ARNs) in der Definition als `"targetArn": "arn:aws:sns:us-east-1:123456789012:underPressureAction"` und angezeigt werden. `"targetArn": "arn:aws:sns:us-east-1:123456789012:pressureClearedAction"`

Weitere Informationen finden Sie im Amazon Simple Notification Service Developer Guide und insbesondere in der Dokumentation des CreateTopicVorgangs in der Amazon Simple Notification Service API-Referenz.

In diesem Beispiel wird auch davon ausgegangen, dass Sie eine AWS Identity and Access Management (IAM-) Rolle mit den entsprechenden Berechtigungen erstellt haben. Der ARN dieser Rolle wird in der Definition des Detektormodells als angezeigt`"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"`. Folgen Sie den Schritten unterEinrichten von Berechtigungen für AWS IoT Events, um diese Rolle zu erstellen, und kopieren Sie den ARN der Rolle an die entsprechende Stelle in der Definition des Detektormodells.

Sie können das Detektormodell mit dem folgenden AWS CLI Befehl erstellen.

```
aws iotevents create-detector-model  --cli-input-json file://motorDetectorModel.json
```

Die Datei `"motorDetectorModel.json"` enthält Folgendes.

```
{
  "detectorModelName": "motorDetectorModel",
```

```
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Normal",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreached",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "Overpressurized",
              "condition": "$input.PressureInput.sensorData.pressure > 70",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreached",
                    "value": "$variable.pressureThresholdBreached + 3"
                  }
                }
              ],
              "nextState": "Dangerous"
            }
          ]
        }
      },
      {
        "stateName": "Dangerous",
        "onEnter": {
          "events": [
            {
              "eventName": "Pressure Threshold Breached",
```

```
              "condition": "$variable.pressureThresholdBreached > 1",
              "actions": [
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "events": [
            {
              "eventName": "Overpressurized",
              "condition": "$input.PressureInput.sensorData.pressure > 70",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreached",
                    "value": "3"
                  }
                }
              ]
            },
            {
              "eventName": "Pressure Okay",
              "condition": "$input.PressureInput.sensorData.pressure <= 70",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "pressureThresholdBreached",
                    "value": "$variable.pressureThresholdBreached - 1"
                  }
                }
              ]
            }
          ],
          "transitionEvents": [
            {
              "eventName": "BackToNormal",
              "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
 $variable.pressureThresholdBreached <= 1",
```

```
              "nextState": "Normal"
            }
          ]
        },
        "onExit": {
          "events": [
            {
              "eventName": "Normal Pressure Restored",
              "condition": "true",
              "actions": [
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
                  }
                }
              ]
            }
          ]
        }
      }
    ],
    "initialStateName": "Normal"
  },
  "key" : "motorid",
  "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}
```

Sendet Nachrichten als Eingaben an einen Detektor

Sie haben jetzt eine Eingabe definiert, die die wichtigen Felder in Nachrichten identifiziert, die von einem Gerät gesendet werden (sieheErstellen Sie einen Eingang zur Erfassung von Gerätedaten). Im vorherigen Abschnitt haben Sie eine erstellt, `detector model` die auf ein Überdruckereignis in einem Motor reagiert (sieheErstellen Sie ein Meldermodell zur Darstellung von Gerätezuständen).

Um das Beispiel zu vervollständigen, senden Sie Nachrichten von einem Gerät (in diesem Fall einem Computer, auf dem der Computer AWS CLI installiert ist) als Eingänge an den Melder.

> **Note**
>
> Wenn Sie ein Meldermodell erstellen oder ein vorhandenes aktualisieren, dauert es einige Minuten, bis das neue oder aktualisierte Meldermodell Nachrichten empfängt und Melder

(Instanzen) erstellt. Wenn Sie das Meldermodell aktualisieren, kann es sein, dass Sie während dieser Zeit weiterhin ein Verhalten beobachten, das auf der vorherigen Version basiert.

Verwenden Sie den folgenden AWS CLI Befehl, um eine Nachricht mit Daten zu senden, die den Schwellenwert überschreiten.

```
aws iotevents-data batch-put-message --cli-input-json file://highPressureMessage.json
 --cli-binary-format raw-in-base64-out
```

Die Datei "`highPressureMessage.json`" enthält Folgendes.

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 80,
 \"temperature\": 39} }"
    }
  ]
}
```

Sie müssen das `messageId` in jeder gesendeten Nachricht ändern. Wenn Sie es nicht ändern, dedupliziert das AWS IoT Events System die Nachrichten. AWS IoT Eventsignoriert eine Nachricht, wenn sie dieselbe enthält `messageID` wie eine andere Nachricht, die innerhalb der letzten fünf Minuten gesendet wurde.

An diesem Punkt wird ein Detektor (Instanz) erstellt, um Ereignisse für den Motor "`Fulton-A32`" zu überwachen. Dieser Detektor wechselt in den "`Normal`" Zustand, in dem er erstellt wurde. Da wir jedoch einen Druckwert über dem Schwellenwert gesendet haben, wechselt er sofort in den "`Dangerous`" Status. Dabei sendet der Detektor eine Nachricht an den Amazon SNS SNS-Endpunkt, dessen ARN lautet`arn:aws:sns:us-east-1:123456789012:underPressureAction`.

Führen Sie den folgenden AWS CLI Befehl aus, um eine Nachricht mit Daten zu senden, die unter dem Druckschwellenwert liegen.

```
aws iotevents-data batch-put-message --cli-input-json file://normalPressureMessage.json
 --cli-binary-format raw-in-base64-out
```

Die Datei `normalPressureMessage.json` enthält Folgendes.

```
{
  "messages": [
    {
      "messageId": "00002",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 60,
 \"temperature\": 29} }"
    }
  ]
}
```

Sie müssen das `messageId` in der Datei jedes Mal ändern, wenn Sie den `BatchPutMessage` Befehl innerhalb von fünf Minuten aufrufen. Senden Sie die Nachricht noch zweimal. Nachdem die Nachricht dreimal gesendet wurde, sendet der Detektor (Instance) für den Motor "`Fulton-A32`" eine Nachricht an den Amazon SNS SNS-Endpunkt "`arn:aws:sns:us-east-1:123456789012:pressureClearedAction`" und wechselt erneut in den "`Normal`" Status.

> **Note**
>
> Sie können mehrere Nachrichten gleichzeitig mit senden. `BatchPutMessage` Die Reihenfolge, in der diese Nachrichten verarbeitet werden, kann jedoch nicht garantiert werden. Um sicherzustellen, dass Nachrichten (Eingaben) in der richtigen Reihenfolge verarbeitet werden, senden Sie sie nacheinander und warten Sie bei jedem API-Aufruf auf eine erfolgreiche Antwort.

Im Folgenden finden Sie Beispiele für Nutzdaten von SNS-Nachrichten, die mit dem in diesem Abschnitt beschriebenen Beispiel für das Detektormodell erstellt wurden.

bei Ereignis „Druckschwellenwert überschritten“

```
IoT> {
```

```
  "eventTime":1558129816420,
  "payload":{
    "actionExecutionId":"5d7444df-a655-3587-a609-dbd7a0f55267",
    "detector":{
      "detectorModelName":"motorDetectorModel",
      "keyValue":"Fulton-A32",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"PressureInput",
      "messageId":"00001",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"Dangerous",
      "variables":{
        "pressureThresholdBreached":3
      },
      "timers":{}
    }
  },
  "eventName":"Pressure Threshold Breached"
}
```

bei Ereignis „Normaler Druck wiederhergestellt“

```
IoT> {
  "eventTime":1558129925568,
  "payload":{
    "actionExecutionId":"7e25fd38-2533-303d-899f-c979792a12cb",
    "detector":{
      "detectorModelName":"motorDetectorModel",
      "keyValue":"Fulton-A32",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"PressureInput",
      "messageId":"00004",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"Dangerous",
      "variables":{
```

```
        "pressureThresholdBreached":0
      },
      "timers":{}
    }
  },
  "eventName":"Normal Pressure Restored"
}
```

Wenn Sie Timer definiert haben, wird deren aktueller Status auch in den Payloads der SNS-Nachrichten angezeigt.

Die Nachrichtennutzdaten enthalten Informationen über den Status des Detektors (Instanz) zum Zeitpunkt des Sendens der Nachricht (d. h. zum Zeitpunkt der Ausführung der SNS-Aktion). Sie können den https://docs.aws.amazon.com/iotevents/latest/apireference/API_iotevents-data_DescribeDetector.html Vorgang verwenden, um ähnliche Informationen über den Status des Melders zu erhalten.

Einschränkungen und Einschränkungen des Detektormodells

Die folgenden Punkte sollten bei der Erstellung eines Detektormodells berücksichtigt werden.

Wie benutzt man das **`actions`** Feld

Das `actions` Feld ist eine Liste von Objekten. Sie können mehr als ein Objekt haben, aber in jedem Objekt ist nur eine Aktion zulässig.

Example

```
            "actions": [
              {
                "setVariable": {
                  "variableName": "pressureThresholdBreached",
                  "value": "$variable.pressureThresholdBreached - 1"
                }
              }
              {
                "setVariable": {
                  "variableName": "temperatureIsTooHigh",
                  "value": "$variable.temperatureIsTooHigh - 1"
                }
              }
```

```
        ]
```

Wie benutzt man das **`condition`** Feld

Das `condition` ist erforderlich für `transitionEvents` und in anderen Fällen optional.

Wenn das `condition` Feld nicht vorhanden ist, entspricht es`"condition": true`.

Das Ergebnis der Auswertung eines Bedingungsausdrucks sollte ein boolescher Wert sein. Wenn das Ergebnis kein boolescher Wert ist, entspricht es dem im Ereignis angegebenen Wert `false` und leitet den Übergang zu dem im `actions` Ereignis `nextState` angegebenen Wert nicht ein.

Verfügbarkeit von Variablenwerten

Wenn der Wert einer Variablen in einem Ereignis festgelegt wird, ist ihr neuer Wert standardmäßig nicht verfügbar oder wird nicht verwendet, um Bedingungen in anderen Ereignissen in derselben Gruppe auszuwerten. Der neue Wert ist nicht verfügbar oder wird in einer Ereignisbedingung im gleichen `onInput` `onExit` Feld `onEnter` oder verwendet.

Stellen Sie den `evaluationMethod` Parameter in der Definition des Detektormodells ein, um dieses Verhalten zu ändern. Wenn der auf gesetzt `evaluationMethod` ist`SERIAL`, werden Variablen aktualisiert und die Ereignisbedingungen werden in der Reihenfolge ausgewertet, in der die Ereignisse definiert sind. Andernfalls werden Variablen innerhalb eines Zustands aktualisiert, und Ereignisse innerhalb eines Zustands werden erst ausgeführt, wenn alle Ereignisbedingungen ausgewertet wurden, wenn für `BATCH` oder standardmäßig dieser Wert festgelegt ist. `evaluationMethod`

Der `"Dangerous"` Status im `onInput` Feld `"$variable.pressureThresholdBreached"` wird um eins dekrementiert, `"Pressure Okay"` falls die Bedingung erfüllt ist (wenn der aktuelle Eingangsdruck kleiner oder gleich 70 ist).

```
{
  "eventName": "Pressure Okay",
  "condition": "$input.PressureInput.sensorData.pressure <= 70",
  "actions": [
    {
      "setVariable": {
        "variableName": "pressureThresholdBreached",
        "value": "$variable.pressureThresholdBreached - 1"
      }
    }
```

```
            ]
          }
```

Der Detektor sollte wieder in den `"Normal"` Zustand zurückkehren, wenn 0 `"$variable.pressureThresholdBreached"` erreicht ist (d. h. wenn der Detektor drei aufeinanderfolgende Druckwerte erhalten hat, die kleiner oder gleich 70 sind). Das `"BackToNormal"` Ereignis `transitionEvents` muss testen, ob der Wert kleiner oder gleich 1 (nicht 0) `"$variable.pressureThresholdBreached"` ist. Außerdem muss erneut überprüft werden, ob der aktuelle Wert, der von gegeben `"$input.PressureInput.sensorData.pressure"` wird, kleiner oder gleich 70 ist.

```
        "transitionEvents": [
          {
            "eventName": "BackToNormal",
            "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreached <= 1",
            "nextState": "Normal"
          }
        ]
```

Andernfalls, wenn die Bedingung nur auf den Wert der Variablen überprüft wird, würden zwei normale Messwerte, gefolgt von einer Überdruckmessung, die Bedingung erfüllen und in den `"Normal"` Zustand zurückkehren. Bei der Bedingung wird der Wert berücksichtigt, der bei der vorherigen Verarbeitung einer Eingabe angegeben `"$variable.pressureThresholdBreached"` wurde. Der Wert der Variablen wird in diesem `"Overpressurized"` Fall auf 3 zurückgesetzt, aber denken Sie daran, dass dieser neue Wert noch für niemanden verfügbar ist`condition`.

Standardmäßig `condition` kann jedes Mal, wenn ein Steuerelement das `onInput` Feld betritt, der Wert einer Variablen nur so sehen, wie er zu Beginn der Verarbeitung der Eingabe war, bevor er durch die unter angegebenen Aktionen geändert wird`onInput`. Das Gleiche gilt für `onEnter` und`onExit`. Jede Änderung, die an einer Variablen vorgenommen wird, wenn wir den Status betreten oder verlassen, ist nicht für andere Bedingungen verfügbar, die in denselben `onEnter` oder `onExit` -Feldern angegeben sind.

Latenz bei der Aktualisierung eines Detektormodells

Wenn Sie ein Detektormodell aktualisieren, löschen und neu erstellen (siehe UpdateDetectorModel), kommt es zu einer gewissen Verzögerung, bis alle generierten Detektoren

(Instanzen) gelöscht werden und das neue Modell zur Neuerstellung der Detektoren verwendet wird. Sie werden neu erstellt, nachdem das neue Detektormodell wirksam wird und neue Eingaben eintreffen. Während dieser Zeit werden Eingaben möglicherweise weiterhin von den Detektoren verarbeitet, die von der vorherigen Version des Detektormodells erzeugt wurden. Während dieses Zeitraums erhalten Sie möglicherweise weiterhin Warnmeldungen, die durch das vorherige Meldermodell definiert wurden.

Leerzeichen in den Eingabetasten

Leerzeichen sind in Eingabeschlüsseln zulässig, aber Verweise auf den Schlüssel müssen in Backticks eingeschlossen werden, und zwar sowohl in der Definition des Eingabeattributs als auch dann, wenn der Wert des Schlüssels in einem Ausdruck referenziert wird. Zum Beispiel bei einer Nachrichtennutzlast wie der folgenden:

```
{
  "motor id": "A32",
  "sensorData" {
    "motor pressure": 56,
    "motor temperature": 39
  }
}
```

Verwenden Sie Folgendes, um die Eingabe zu definieren.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.`motor pressure`" },
      { "jsonPath": "`motor id`" }
    ]
  }
}
```

In einem bedingten Ausdruck müssen Sie auch mithilfe von Backticks auf den Wert eines solchen Schlüssels verweisen.

```
$input.PressureInput.sensorData.`motor pressure`
```

Ein kommentiertes Beispiel: HVAC-Temperatursteuerung

Einige der folgenden JSON-Beispieldateien enthalten eingebettete Kommentare, was sie zu ungültigen JSON-Dateien macht. Vollständige Versionen dieser Beispiele ohne Kommentare finden Sie unterHVAC-Temperatursteuerung.

Hintergrund

In diesem Beispiel wird ein Thermostatsteuerungsmodell implementiert, das Ihnen folgende Möglichkeiten bietet.

- Definieren Sie nur ein Meldermodell, das zur Überwachung und Steuerung mehrerer Bereiche verwendet werden kann. Für jeden Bereich wird eine Melderinstanz erstellt.
- Erfassen Sie Temperaturdaten von mehreren Sensoren in jedem Kontrollbereich.
- Ändern Sie den Temperatursollwert für einen Bereich.
- Stellen Sie die Betriebsparameter für jeden Bereich ein und setzen Sie diese Parameter zurück, während die Instanz verwendet wird.
- Dynamisches Hinzufügen oder Löschen von Sensoren aus einem Bereich.
- Geben Sie eine Mindestlaufzeit an, um Heiz- und Kühlgeräte zu schützen.
- Lehnen Sie anomale Sensormesswerte ab.
- Definieren Sie Notfall-Sollwerte, die sofort Heizen oder Kühlen einschalten, wenn ein Sensor eine Temperatur über oder unter einem bestimmten Schwellenwert meldet.
- Melden Sie anomale Messwerte und Temperaturspitzen.

Definitionen eingeben

Wir wollen ein Detektormodell erstellen, mit dem wir die Temperatur in verschiedenen Bereichen überwachen und steuern können. Jeder Bereich kann mehrere Sensoren haben, die die Temperatur melden. Wir gehen davon aus, dass jeder Bereich von einer Heizeinheit und einer Kühleinheit versorgt wird, die ein- oder ausgeschaltet werden können, um die Temperatur in dem Bereich zu regeln. Jeder Bereich wird von einer Melderinstanz gesteuert.

Da die verschiedenen Bereiche, die wir überwachen und steuern, unterschiedliche Eigenschaften aufweisen können, die unterschiedliche Steuerparameter erfordern, definieren wir die `'seedTemperatureInput'` so, dass diese Parameter für jeden Bereich bereitgestellt

werden. Wenn wir eine dieser Eingangsnachrichten an sendenAWS IoT Events, wird eine neue Detektormodellinstanz erstellt, die die Parameter enthält, die wir in diesem Bereich verwenden möchten. Hier ist die Definition dieser Eingabe.

CLI-Befehl:

```
aws iotevents create-input --cli-input-json file://seedInput.json
```

Datei: `seedInput.json`

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "areaId" },
      { "jsonPath": "desiredTemperature" },
      { "jsonPath": "allowedError" },
      { "jsonPath": "rangeHigh" },
      { "jsonPath": "rangeLow" },
      { "jsonPath": "anomalousHigh" },
      { "jsonPath": "anomalousLow" },
      { "jsonPath": "sensorCount" },
      { "jsonPath": "noDelay" }
    ]
  }
}
```

Antwort:

```
{
    "inputConfiguration": {
        "status": "ACTIVE",
        "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/
seedTemperatureInput",
        "lastUpdateTime": 1557519620.736,
        "creationTime": 1557519620.736,
        "inputName": "seedTemperatureInput",
        "inputDescription": "Temperature seed values."
    }
}
```

Hinweise

- Für jedes in einer Nachricht `'areaId'` empfangene Unikat wird eine neue Detektorinstanz erstellt. Sehen Sie sich das `'key'` Feld in der `'areaDetectorModel'` Definition an.
- Die Durchschnittstemperatur kann `'desiredTemperature'` von dem abweichen, `'allowedError'` bevor die Heiz- oder Kühlgeräte für den Bereich aktiviert werden.
- Meldet ein Sensor eine Temperatur über dem`'rangeHigh'`, meldet der Melder einen Temperaturanstieg und startet sofort die Kühleinheit.
- Meldet ein Sensor eine Temperatur unter dem`'rangeLow'`, meldet der Melder einen Temperaturanstieg und startet sofort die Heizeinheit.
- Wenn ein Sensor eine Temperatur über `'anomalousHigh'` oder unter dem Wert meldet`'anomalousLow'`, meldet der Melder einen anomalen Sensorwert, ignoriert jedoch den gemeldeten Temperaturwert.
- Das `'sensorCount'` teilt dem Melder mit, wie viele Sensoren für den Bereich Bericht erstatten. Der Detektor berechnet die Durchschnittstemperatur in dem Gebiet, indem er jedem Temperaturmesswert, den er empfängt, den entsprechenden Gewichtungsfaktor zuweist. Aus diesem Grund muss der Detektor nicht nachverfolgen, was jeder Sensor meldet, und die Anzahl der Sensoren kann je nach Bedarf dynamisch geändert werden. Wenn jedoch ein einzelner Sensor offline geht, weiß der Melder dies nicht und berücksichtigt es auch nicht. Wir empfehlen Ihnen, ein anderes Meldermodell speziell für die Überwachung des Verbindungsstatus der einzelnen Sensoren zu erstellen. Zwei sich ergänzende Detektormodelle vereinfachen das Design beider.
- Der `'noDelay'` Wert kann `true` oder sein`false`. Nach dem Einschalten eines Heiz- oder Kühlgeräts sollte es für eine gewisse Mindestzeit eingeschaltet bleiben, um die Integrität des Geräts zu schützen und seine Lebensdauer zu verlängern. Wenn auf eingestellt `'noDelay'` ist`false`, erzwingt die Melderinstanz eine Verzögerung, bevor sie die Kühl- und Heizgeräte ausschaltet, um sicherzustellen, dass sie so lange wie möglich laufen. Die Anzahl der Sekunden der Verzögerung wurde in der Definition des Detektormodells fest codiert, da wir keinen Variablenwert verwenden können, um einen Timer einzustellen.

Der `'temperatureInput'` wird verwendet, um Sensordaten an eine Detektorinstanz zu übertragen.

CLI-Befehl:

```
aws iotevents create-input --cli-input-json file://temperatureInput.json
```

Datei: `temperatureInput.json`

```
{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorId" },
      { "jsonPath": "areaId" },
      { "jsonPath": "sensorData.temperature" }
    ]
  }
}
```

Antwort:

```
{
    "inputConfiguration": {
        "status": "ACTIVE",
        "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/temperatureInput",
        "lastUpdateTime": 1557519707.399,
        "creationTime": 1557519707.399,
        "inputName": "temperatureInput",
        "inputDescription": "Temperature sensor unit data."
    }
}
```

Hinweise

- Die wird `'sensorId'` nicht von einer Beispiel-Detektorinstanz verwendet, um einen Sensor direkt zu steuern oder zu überwachen. Es wird automatisch an Benachrichtigungen weitergegeben, die von der Detektorinstanz gesendet werden. Von dort aus kann es verwendet werden, um die Sensoren zu identifizieren, die ausfallen (z. B. könnte ein Sensor, der regelmäßig ungewöhnliche Messwerte sendet, bald ausfallen) oder die offline gegangen sind (wenn er als Eingang für ein zusätzliches Meldermodell verwendet wird, das den Herzschlag des Geräts überwacht). `'sensorId'`Sie können auch dabei helfen, warme oder kalte Zonen in einem Gebiet zu identifizieren, wenn die Messwerte regelmäßig vom Durchschnitt abweichen.
- Das `'areaId'` wird verwendet, um die Daten des Sensors an die entsprechende Melderinstanz weiterzuleiten. Für jedes in einer Nachricht `'areaId'` empfangene Unikat wird eine

Detektorinstanz erstellt. Sehen Sie sich das `'key'` Feld in der `'areaDetectorModel'` Definition an.

Definition des Detektormodells

Das `'areaDetectorModel'` Beispiel enthält Kommentare in der Zeile.

CLI-Befehl:

```
aws iotevents create-detector-model --cli-input-json file://areaDetectorModel.json
```

Datei: `areaDetectorModel.json`

```
{
  "detectorModelName": "areaDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "start",
        // In the 'start' state we set up the operation parameters of the new detector
 instance.
        //   We get here when the first input message arrives. If that is a
 'seedTemperatureInput'
        //   message, we save the operation parameters, then transition to the 'idle'
 state. If
        //   the first message is a 'temperatureInput', we wait here until we get a
        //   'seedTemperatureInput' input to ensure our operation parameters are set.
 We can
        //   also reenter this state using the 'BatchUpdateDetector' API. This enables
 us to
        //   reset the operation parameters without needing to delete the detector
 instance.
        "onEnter": {
          "events": [
            {
              "eventName": "prepare",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
```

```
                    // initialize 'sensorId' to an invalid value (0) until an actual
sensor reading
                    //   arrives
                    "variableName": "sensorId",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    // initialize 'reportedTemperature' to an invalid value (0.1) until
an actual
                    //   sensor reading arrives
                    "variableName": "reportedTemperature",
                    "value": "0.1"
                  }
                },
                {
                  "setVariable": {
                    // When using 'BatchUpdateDetector' to re-enter this state, this
variable should
                    //   be set to true.
                    "variableName": "resetMe",
                    "value": "false"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "initialize",
              "condition": "$input.seedTemperatureInput.sensorCount > 0",
              // When a 'seedTemperatureInput' message with a valid 'sensorCount' is
received,
              //   we use it to set the operational parameters for the area to be
monitored.
              "actions": [
                {
                  "setVariable": {
                    "variableName": "rangeHigh",
                    "value": "$input.seedTemperatureInput.rangeHigh"
                  }
```

```
                },
                {
                  "setVariable": {
                    "variableName": "rangeLow",
                    "value": "$input.seedTemperatureInput.rangeLow"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "desiredTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                },
                {
                  "setVariable": {
                    // Assume we're at the desired temperature when we start.
                    "variableName": "averageTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "allowedError",
                    "value": "$input.seedTemperatureInput.allowedError"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "anomalousHigh",
                    "value": "$input.seedTemperatureInput.anomalousHigh"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "anomalousLow",
                    "value": "$input.seedTemperatureInput.anomalousLow"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "sensorCount",
                    "value": "$input.seedTemperatureInput.sensorCount"
                  }
                },
```

```
              {
                "setVariable": {
                  "variableName": "noDelay",
                  "value": "$input.seedTemperatureInput.noDelay == true"
                }
              }
            ],
            "nextState": "idle"
          },
          {
            "eventName": "reset",
            "condition": "($variable.resetMe == true) &&
($input.temperatureInput.sensorData.temperature < $variable.anomalousHigh &&
$input.temperatureInput.sensorData.temperature > $variable.anomalousLow)",
            // This event is triggered if we have reentered the 'start' state using
the
            //   'BatchUpdateDetector' API with 'resetMe' set to true. When we
reenter using
            //   'BatchUpdateDetector' we do not automatically continue to the 'idle'
state, but
            //   wait in 'start' until the next input message arrives. This event
enables us to
            //   transition to 'idle' on the next valid 'temperatureInput' message
that arrives.
            "actions": [
              {
                "setVariable": {
                  "variableName": "averageTemperature",
                  "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
              }
            ],
            "nextState": "idle"
          }
        ]
      },
      "onExit": {
        "events": [
          {
            "eventName": "resetHeatCool",
            "condition": "true",
            // Make sure the heating and cooling units are off before entering
'idle'.
```

```
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Heating/Off"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Cooling/Off"
                }
              }
            ]
          }
        ]
      }
    },


    {
      "stateName": "idle",
      "onInput": {
        "events": [
          {
            "eventName": "whatWasInput",
            "condition": "true",
            // By storing the 'sensorId' and the 'temperature' in variables, we make
them
            //   available in any messages we send out to report anomalies, spikes,
or just
            //   if needed for debugging.
            "actions": [
              {
                "setVariable": {
                  "variableName": "sensorId",
```

```
                    "value": "$input.temperatureInput.sensorId"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "reportedTemperature",
                    "value": "$input.temperatureInput.sensorData.temperature"
                  }
                }
              ]
            },
            {
              "eventName": "changeDesired",
              "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
              // This event enables us to change the desired temperature at any time by
sending a
              //   'seedTemperatureInput' message. But note that other operational
parameters are not
              //   read or changed.
              "actions": [
                {
                  "setVariable": {
                    "variableName": "desiredTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                }
              ]
            },
            {
              "eventName": "calculateAverage",
              "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
              // If a valid temperature reading arrives, we use it to update the
average temperature.
              //   For simplicity, we assume our sensors will be sending updates at
about the same rate,
              //   so we can calculate an approximate average by giving equal weight to
each reading we receive.
              "actions": [
                {
                  "setVariable": {
                    "variableName": "averageTemperature",
```

```
                    "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                  }
                }
              ]
            }
          ],
          "transitionEvents": [
            {
              "eventName": "anomalousInputArrived",
              "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
              // When an anomalous reading arrives, send an MQTT message, but stay in
the current state.
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/anomaly"
                  }
                }
              ],
              "nextState": "idle"
            },

            {
              "eventName": "highTemperatureSpike",
              "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
              // When even a single temperature reading arrives that is above the
'rangeHigh', take
              //   emergency action to begin cooling, and report a high temperature
spike.
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                  }
                },
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
                  }
                },
```

```
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/On"
                  }
                },
                {
                  "setVariable": {
                    // This is necessary because we want to set a timer to delay the
shutoff
                    //   of a cooling/heating unit, but we only want to set the timer
when we
                    //   enter that new state initially.
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "cooling"
            },

            {
              "eventName": "lowTemperatureSpike",
              "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
              // When even a single temperature reading arrives that is below the
'rangeLow', take
              //   emergency action to begin heating, and report a low-temperature
spike.
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                  }
                },
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                  }
                },
```

```
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "heating"
            },

            {
              "eventName": "highTemperatureThreshold",
              "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >
($variable.desiredTemperature + $variable.allowedError))",
              // When the average temperature is above the desired temperature plus the
allowed error factor,
              //   it is time to start cooling. Note that we calculate the average
temperature here again
              //   because the value stored in the 'averageTemperature' variable is not
yet available for use
              //   in our condition.
              "actions": [
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/On"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "cooling"
            },

            {
```

```
            "eventName": "lowTemperatureThreshold",
            "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <
($variable.desiredTemperature - $variable.allowedError))",
            // When the average temperature is below the desired temperature minus
the allowed error factor,
            //   it is time to start heating. Note that we calculate the average
temperature here again
            //   because the value stored in the 'averageTemperature' variable is not
yet available for use
            //   in our condition.
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Heating/On"
                }
              },
              {
                "setVariable": {
                  "variableName": "enteringNewState",
                  "value": "true"
                }
              }
            ],
            "nextState": "heating"
          }
        ]
      }
    },


    {
      "stateName": "cooling",
      "onEnter": {
        "events": [
          {
            "eventName": "delay",
            "condition": "!$variable.noDelay && $variable.enteringNewState",
```

```
              // If the operational parameters specify that there should be a minimum
time that the
              //   heating and cooling units should be run before being shut off again,
we set
              //   a timer to ensure the proper operation here.
              "actions": [
                {
                  "setTimer": {
                    "timerName": "coolingTimer",
                    "seconds": 180
                  }
                },
                {
                  "setVariable": {
                    // We use this 'goodToGo' variable to store the status of the timer
expiration
                    //   for use in conditions that also use input variable values. If
                    //   'timeout()' is used in such mixed conditionals, its value is
lost.
                    "variableName": "goodToGo",
                    "value": "false"
                  }
                }
              ]
            },
            {
              "eventName": "dontDelay",
              "condition": "$variable.noDelay == true",
              // If the heating/cooling unit shutoff delay is not used, no need to
wait.
              "actions": [
                {
                  "setVariable": {
                    "variableName": "goodToGo",
                    "value": "true"
                  }
                }
              ]
            },
            {
              "eventName": "beenHere",
              "condition": "true",
              "actions": [
                {
```

```
                "setVariable": {
                  "variableName": "enteringNewState",
                  "value": "false"
                }
              }
            ]
          }
        ]
      },

      "onInput": {
        "events": [
          // These are events that occur when an input is received (if the condition
is
          //   satisfied), but don't cause a transition to another state.
          {
            "eventName": "whatWasInput",
            "condition": "true",
            "actions": [
              {
                "setVariable": {
                  "variableName": "sensorId",
                  "value": "$input.temperatureInput.sensorId"
                }
              },
              {
                "setVariable": {
                  "variableName": "reportedTemperature",
                  "value": "$input.temperatureInput.sensorData.temperature"
                }
              }
            ]
          },
          {
            "eventName": "changeDesired",
            "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
            "actions": [
              {
                "setVariable": {
                  "variableName": "desiredTemperature",
                  "value": "$input.seedTemperatureInput.desiredTemperature"
                }
              }
```

```
            ]
          },
          {
            "eventName": "calculateAverage",
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
            "actions": [
              {
                "setVariable": {
                  "variableName": "averageTemperature",
                  "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
              }
            ]
          },
          {
            "eventName": "areWeThereYet",
            "condition": "(timeout(\"coolingTimer\"))",
            "actions": [
              {
                "setVariable": {
                  "variableName": "goodToGo",
                  "value": "true"
                }
              }
            ]
          }
        ],
        "transitionEvents": [
          // Note that some tests of temperature values (for example, the test for an
anomalous value)
          //   must be placed here in the 'transitionEvents' because they work
together with the tests
          //   in the other conditions to ensure that we implement the proper
"if..elseif..else" logic.
          //   But each transition event must have a destination state ('nextState'),
and even if that
          //   is actually the current state, the "onEnter" events for this state
will be executed again.
          //   This is the reason for the 'enteringNewState' variable and related.
          {
            "eventName": "anomalousInputArrived",
```

```
            "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/anomaly"
                }
              }
            ],
            "nextState": "cooling"
          },

          {
            "eventName": "highTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/spike"
                }
              }
            ],
            "nextState": "cooling"
          },

          {
            "eventName": "lowTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/spike"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
                }
              },
              {
                "sns": {
```

```
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/Off"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "heating"
            },

            {
              "eventName": "desiredTemperature",
              "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <=
($variable.desiredTemperature - $variable.allowedError)) && $variable.goodToGo ==
true",
              "actions": [
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/Off"
                  }
                }
              ],
              "nextState": "idle"
            }
          ]
```

```
        }
      },


      {
        "stateName": "heating",
        "onEnter": {
          "events": [
            {
              "eventName": "delay",
              "condition": "!$variable.noDelay && $variable.enteringNewState",
              "actions": [
                {
                  "setTimer": {
                    "timerName": "heatingTimer",
                    "seconds": 120
                  }
                },
                {
                  "setVariable": {
                    "variableName": "goodToGo",
                    "value": "false"
                  }
                }
              ]
            },
            {
              "eventName": "dontDelay",
              "condition": "$variable.noDelay == true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "goodToGo",
                    "value": "true"
                  }
                }
              ]
            },
            {
              "eventName": "beenHere",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
```

```
                    "variableName": "enteringNewState",
                    "value": "false"
                  }
                }
              ]
            }
          ]
        },

        "onInput": {
          "events": [
            {
              "eventName": "whatWasInput",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "sensorId",
                    "value": "$input.temperatureInput.sensorId"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "reportedTemperature",
                    "value": "$input.temperatureInput.sensorData.temperature"
                  }
                }
              ]
            },
            {
              "eventName": "changeDesired",
              "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "desiredTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                }
              ]
            },
            {
              "eventName": "calculateAverage",
```

```
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
            "actions": [
              {
                "setVariable": {
                  "variableName": "averageTemperature",
                  "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
              }
            ]
          },
          {
            "eventName": "areWeThereYet",
            "condition": "(timeout(\"heatingTimer\"))",
            "actions": [
              {
                "setVariable": {
                  "variableName": "goodToGo",
                  "value": "true"
                }
              }
            ]
          }
        ],
        "transitionEvents": [
          {
            "eventName": "anomalousInputArrived",
            "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/anomaly"
                }
              }
            ],
            "nextState": "heating"
          },

          {
            "eventName": "highTemperatureSpike",
```

```
            "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/spike"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Heating/Off"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Cooling/On"
                }
              },
              {
                "setVariable": {
                  "variableName": "enteringNewState",
                  "value": "true"
                }
              }
            ],
            "nextState": "cooling"
          },

          {
            "eventName": "lowTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
            "actions": [
              {
```

```
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/spike"
                }
              }
            ],
            "nextState": "heating"
          },

          {
            "eventName": "desiredTemperature",
            "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >=
($variable.desiredTemperature + $variable.allowedError)) && $variable.goodToGo ==
true",
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Heating/Off"
                }
              }
            ],
            "nextState": "idle"
          }
        ]
      }
    }

  ],

  "initialStateName": "start"
 },
 "key": "areaId",
 "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}
```

Antwort:

```
{
```

```
    "detectorModelConfiguration": {
        "status": "ACTIVATING",
        "lastUpdateTime": 1557523491.168,
        "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
        "creationTime": 1557523491.168,
        "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
areaDetectorModel",
        "key": "areaId",
        "detectorModelName": "areaDetectorModel",
        "detectorModelVersion": "1"
    }
}
```

BatchUpdateDetectorBeispiel

Sie können die `BatchUpdateDetector` Operation verwenden, um eine Detektorinstanz in einen bekannten Zustand zu versetzen, einschließlich Timer- und Variablenwerten. Im folgenden Beispiel werden die `BatchUpdateDetector` Betriebsparameter für einen Bereich zurückgesetzt, der unter Temperaturüberwachung und -steuerung steht. Mit diesem Vorgang können Sie dies tun, ohne das Meldermodell löschen, neu erstellen oder aktualisieren zu müssen.

CLI-Befehl:

```
aws iotevents-data batch-update-detector --cli-input-json file://areaDM.BUD.json
```

Datei: `areaDM.BUD.json`

```
{
  "detectors": [
    {
      "messageId": "0001",
      "detectorModelName": "areaDetectorModel",
      "keyValue": "Area51",
      "state": {
        "stateName": "start",
        "variables": [
          {
            "name": "desiredTemperature",
            "value": "22"
          },
          {
```

```
            "name": "averageTemperature",
            "value": "22"
          },
          {
            "name": "allowedError",
            "value": "1.0"
          },
          {
            "name": "rangeHigh",
            "value": "30.0"
          },
          {
            "name": "rangeLow",
            "value": "15.0"
          },
          {
            "name": "anomalousHigh",
            "value": "60.0"
          },
          {
            "name": "anomalousLow",
            "value": "0.0"
          },
          {
            "name": "sensorCount",
            "value": "12"
          },
          {
            "name": "noDelay",
            "value": "true"
          },
          {
            "name": "goodToGo",
            "value": "true"
          },
          {
            "name": "sensorId",
            "value": "0"
          },
          {
            "name": "reportedTemperature",
            "value": "0.1"
          },
          {
```

```
            "name": "resetMe",
            // When 'resetMe' is true, our detector model knows that we have reentered
 the 'start' state
            //   to reset operational parameters, and will allow the next valid
 temperature sensor
            //   reading to cause the transition to the 'idle' state.
            "value": "true"
          }
        ],
        "timers": [
        ]
      }
    }
  ]
}
```

Antwort:

```
{
    "batchUpdateDetectorErrorEntries": []
}
```

BatchPutMessageBeispiele

Example 1

Verwenden Sie den `BatchPutMessage` Vorgang, um eine `"seedTemperatureInput"` Nachricht zu senden, in der die Betriebsparameter für einen bestimmten Bereich festgelegt werden, für den die Temperatur geregelt und überwacht wird. Jede NachrichtAWS IoT Events, die von dieser empfangen wird, `"areaId"` hat eine neue Melderinstanz zur Folge. Die neue Melderinstanz ändert ihren Status jedoch nicht `"idle"` und beginnt erst, die Temperatur zu überwachen und Heiz- oder Kühlgeräte zu steuern, wenn eine `"seedTemperatureInput"` Meldung für den neuen Bereich eingeht.

CLI-Befehl:

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json --cli-
binary-format raw-in-base64-out
```

Datei: `seedExample.json`

```
{
```

```
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 20.0, \"allowedError
\": 0.7, \"rangeHigh\": 30.0, \"rangeLow\": 15.0, \"anomalousHigh\": 60.0,
 \"anomalousLow\": 0.0, \"sensorCount\": 10, \"noDelay\": false}"
    }
  ]
}
```

Antwort:

```
{
    "BatchPutMessageErrorEntries": []
}
```

Example

2

Verwenden Sie den `BatchPutMessage` Vorgang, um eine `"temperatureInput"` Nachricht zu senden, um Temperatursensordaten für einen Sensor in einem bestimmten Steuerungs- und Überwachungsbereich zu melden.

CLI-Befehl:

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json --
cli-binary-format raw-in-base64-out
```

Datei: `temperatureExample.json`

```
{
  "messages": [
    {
      "messageId": "00005",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"05\", \"areaId\": \"Area51\", \"sensorData\":
 {\"temperature\": 23.12} }"
    }
  ]
```

```
}
```

Antwort:

```
{
    "BatchPutMessageErrorEntries": []
}
```

Example 3

Verwenden Sie den `BatchPutMessage` Vorgang, um eine `"seedTemperatureInput"` Nachricht zu senden, um den Wert der gewünschten Temperatur für einen bestimmten Bereich zu ändern.

CLI-Befehl:

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json --
cli-binary-format raw-in-base64-out
```

Datei: `seedSetDesiredTemp.json`

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

Antwort:

```
{
    "BatchPutMessageErrorEntries": []
}
```

Beispiel: Aufnahme von MQTT-Nachrichten

Wenn Ihre Sensor-Computing-Ressourcen die `"BatchPutMessage"` API nicht verwenden können, aber ihre Daten mit einem einfachen MQTT-Client an den AWS IoT Core Message Broker senden

können, können Sie eine AWS IoT Core Themenregel erstellen, um Nachrichtendaten an eine Eingabe umzuleiten. AWS IoT Events Im Folgenden finden Sie eine Definition einer AWS IoT Events Themenregel, die die Eingabefelder `"areaId"` und die `"sensorId"` Eingabefelder aus dem MQTT-Thema und das `"sensorData.temperature"` Feld aus dem `"temp"` Nachrichten-Payload-Feld verwendet und diese Daten in unsere aufnimmt. AWS IoT Events `"temperatureInput"`

Wenn Ihre Sensor-Computing-Ressourcen die `"BatchPutMessage"` API nicht verwenden können, aber ihre Daten mit einem einfachen MQTT-Client an den AWS IoT Core Message Broker senden können, können Sie eine AWS IoT Core Themenregel erstellen, um Nachrichtendaten an eine Eingabe umzuleiten. AWS IoT Events Im Folgenden finden Sie eine Definition einer AWS IoT Events Themenregel, die die Eingabefelder `"areaId"` und die `"sensorId"` Eingabefelder aus dem MQTT-Thema und das `"sensorData.temperature"` Feld aus dem `"temp"` Nachrichten-Payload-Feld verwendet und diese Daten in unsere aufnimmt. AWS IoT Events `"temperatureInput"`

CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://temperatureTopicRule.json
```

Datei: `seedSetDesiredTemp.json`

```
{
  "ruleName": "temperatureTopicRule",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as areaId, topic(4) as sensorId, temp as
 sensorData.temperature FROM 'update/temperature/#'",
    "description": "Ingest temperature sensor messages into IoT Events",
    "actions": [
      {
        "iotEvents": {
          "inputName": "temperatureInput",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/anotheRole"
        }
      }
    ],
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23"
  }
}
```

Antwort: [keine]

Wenn der Sensor eine Nachricht zum Thema "update/temperature/Area51/03" mit der folgenden Nutzlast sendet.

```
{ "temp": 24.5 }
```

Dies führt dazu, dass Daten aufgenommen werden, AWS IoT Events als ob der folgende "BatchPutMessage" API-Aufruf getätigt worden wäre.

```
aws iotevents-data batch-put-message --cli-input-json file://spoofExample.json --cli-
binary-format raw-in-base64-out
```

Datei: spoofExample.json

```
{
  "messages": [
    {
      "messageId": "54321",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"03\", \"areaId\": \"Area51\", \"sensorData\":
 {\"temperature\": 24.5} }"
    }
  ]
}
```

Beispiele: Generierte Amazon SNS SNS-Nachrichten

Im Folgenden finden Sie Beispiele für SNS-Nachrichten, die von der "Area51" Detector-Instance generiert wurden.

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{

 "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},"event
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature"
{}}},"eventName":"resetHeatCool"}
```

```
Cooling system off command> {"eventTime":1557520274729,"payload":
{"actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192","detector":
{"detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},"event
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature"
{}}},"eventName":"resetHeatCool"}
```

Beispiel: API DescribeDetector

Sie können den `DescribeDetector` Vorgang verwenden, um den aktuellen Status, die Variablenwerte und die Timer für eine Detektorinstanz anzuzeigen.

CLI-Befehl:

```
aws iotevents-data describe-detector --detector-model-name areaDetectorModel --key-
value Area51
```

Antwort:

```
{
    "detector": {
        "lastUpdateTime": 1557521572.216,
        "creationTime": 1557520274.405,
        "state": {
            "variables": [
                {
                    "name": "resetMe",
                    "value": "false"
                },
                {
                    "name": "rangeLow",
                    "value": "15.0"
                },
                {
                    "name": "noDelay",
                    "value": "false"
                },
                {
                    "name": "desiredTemperature",
```

```
                    "value": "20.0"
                },
                {
                    "name": "anomalousLow",
                    "value": "0.0"
                },
                {
                    "name": "sensorId",
                    "value": "\"01\""
                },
                {
                    "name": "sensorCount",
                    "value": "10"
                },
                {
                    "name": "rangeHigh",
                    "value": "30.0"
                },
                {
                    "name": "enteringNewState",
                    "value": "false"
                },
                {
                    "name": "averageTemperature",
                    "value": "19.572"
                },
                {
                    "name": "allowedError",
                    "value": "0.7"
                },
                {
                    "name": "anomalousHigh",
                    "value": "60.0"
                },
                {
                    "name": "reportedTemperature",
                    "value": "15.72"
                },
                {
                    "name": "goodToGo",
                    "value": "false"
                }
            ],
            "stateName": "idle",
```

```
            "timers": [
                {
                    "timestamp": 1557520454.0,
                    "name": "idleTimer"
                }
            ]
        },
        "keyValue": "Area51",
        "detectorModelName": "areaDetectorModel",
        "detectorModelVersion": "1"
    }
}
```

AWS IoT CoreBeispiele für die Regel-Engine

Die folgenden Regeln veröffentlichen AWS IoT Core MQTT-Nachrichten erneut als Shadow-Update-Anforderungsnachrichten. Wir gehen davon aus, dass für jeden Bereich, der durch das Detektormodell gesteuert wird, AWS IoT Core Dinge für eine Heiz- und eine Kühleinheit definiert sind. In diesem Beispiel haben wir Dinge mit dem Namen `"Area51HeatingUnit"` und definiert`"Area51CoolingUnit"`.

CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOffRule.json
```

Datei: `ADMShadowCoolOffRule.json`

```
{
  "ruleName": "ADMShadowCoolOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/Off'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
```

```
        }
      }
    ]
  }
}
```

Antwort: [leer]

CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOnRule.json
```

Datei: ADMShadowCoolOnRule.json

```
{
  "ruleName": "ADMShadowCoolOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/On'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowHeatOffRule.json
```

Datei: ADMShadowHeatOffRule.json

```
{
  "ruleName": "ADMShadowHeatOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/Off'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowHeatOnRule.json
```

Datei: ADMShadowHeatOnRule.json

```
{
  "ruleName": "ADMShadowHeatOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/On'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
```

```
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

Unterstützte Aktionen

AWS IoT Events kann Aktionen auslösen, wenn es ein bestimmtes Ereignis oder Übergangsereignis erkennt. Sie können integrierte Aktionen definieren, um einen Timer zu verwenden, eine Variable festzulegen oder Daten an andere AWS Ressourcen zu senden.

> **Note**
>
> Wenn Sie eine Aktion in einem Detektormodell definieren, können Sie Ausdrücke für Parameter verwenden, die vom Datentyp Zeichenfolge sind. Weitere Informationen finden Sie unter Ausdrücke.

AWS IoT Events unterstützt die folgenden Aktionen, mit denen Sie einen Timer verwenden oder eine Variable festlegen können:

- `setTimer` , um einen Timer zu erstellen.
- `resetTimer` , um den Timer zurückzusetzen.
- `clearTimer` , um den Timer zu löschen.
- `setVariable` , um eine Variable zu erstellen.

AWS IoT Events unterstützt die folgenden Aktionen, mit denen Sie mit - AWS Services arbeiten können:

- `iotTopicPublish` , um eine Nachricht zu einem MQTT-Thema zu veröffentlichen.
- `iotEvents` , um Daten AWS IoT Events als Eingabewert an zu senden.
- `iotSiteWise` zum Senden von Daten an eine Komponenteneigenschaft in AWS IoT SiteWise
- `dynamoDB` , um Daten an eine Amazon-DynamoDB-Tabelle zu senden.
- `dynamoDBv2` , um Daten an eine Amazon-DynamoDB-Tabelle zu senden.
- `firehose` , um Daten an einen Amazon-Data-Firehose-Stream zu senden.
- `lambda` , um eine AWS Lambda -Funktion aufzurufen.
- `sns` , um Daten als Push-Benachrichtigung zu senden.
- `sqs` , um Daten an eine Amazon SQS-Warteschlange zu senden.

Verwenden von integrierten Aktionen

AWS IoT Events unterstützt die folgenden Aktionen, mit denen Sie einen Timer verwenden oder eine Variable festlegen können:

- `setTimer` , um einen Timer zu erstellen.
- `resetTimer` , um den Timer zurückzusetzen.
- `clearTimer` , um den Timer zu löschen.
- `setVariable` , um eine Variable zu erstellen.

Timer-Aktion festlegen

Set timer action

Mit der `setTimer` Aktion können Sie einen Timer mit einer Dauer in Sekunden erstellen.

More information (2)

Wenn Sie einen Timer erstellen, müssen Sie die folgenden Parameter angeben.

`timerName`

Der Name des Timers.

`durationExpression`

(Optional) Die Dauer des Timers in Sekunden.

Das ausgewertete Ergebnis eines Dauerausdrucks wird auf die nächste ganze Zahl abgerundet. Wenn Sie beispielsweise den Timer auf 60,99 Sekunden festlegen, beträgt das ausgewertete Ergebnis des Dauerausdrucks 60 Sekunden.

Weitere Informationen finden Sie unter SetTimerAction in der AWS IoT Events -API-Referenz.

Timer-Aktion zurücksetzen

Reset timer action

Mit der `resetTimer` Aktion können Sie den Timer auf das zuvor ausgewertete Ergebnis des Dauerausdrucks festlegen.

More information (1)

Wenn Sie einen Timer zurücksetzen, müssen Sie den folgenden Parameter angeben.

`timerName`

Der Name des Timers.

AWS IoT Events bewertet den Dauerausdruck nicht erneut, wenn Sie den Timer zurücksetzen.

Weitere Informationen finden Sie unter ResetTimerAction in der AWS IoT Events -API-Referenz.

Timer-Aktion löschen

Clear timer action

Mit der `clearTimer` Aktion können Sie einen vorhandenen Timer löschen.

More information (1)

Wenn Sie einen Timer löschen, müssen Sie den folgenden Parameter angeben.

`timerName`

Der Name des Timers.

Weitere Informationen finden Sie unter ClearTimerAction in der AWS IoT Events -API-Referenz.

Festlegen einer Variablenaktion

Set variable action

Mit der `setVariable` Aktion können Sie eine Variable mit einem angegebenen Wert erstellen.

More information (2)

Wenn Sie eine Variable erstellen, müssen Sie die folgenden Parameter angeben.

`variableName`

Der Name der Variable.

`value`

Der neue Wert der Variable.

Weitere Informationen finden Sie unter SetVariableAction in der AWS IoT Events -API-Referenz.

Arbeiten mit anderen - AWS Services

AWS IoT Events unterstützt die folgenden Aktionen, mit denen Sie mit - AWS Services arbeiten können:

- `iotTopicPublish` , um eine Nachricht zu einem MQTT-Thema zu veröffentlichen.
- `iotEvents` , um Daten AWS IoT Events als Eingabewert an zu senden.
- `iotSiteWise` zum Senden von Daten an eine Komponenteneigenschaft in AWS IoT SiteWise
- `dynamoDB` , um Daten an eine Amazon-DynamoDB-Tabelle zu senden.
- `dynamoDBv2` , um Daten an eine Amazon-DynamoDB-Tabelle zu senden.
- `firehose` , um Daten an einen Amazon-Data-Firehose-Stream zu senden.
- `lambda` , um eine AWS Lambda -Funktion aufzurufen.
- `sns` , um Daten als Push-Benachrichtigung zu senden.
- `sqs` , um Daten an eine Amazon SQS-Warteschlange zu senden.

> **Important**
>
> - Sie müssen dieselbe AWS Region für sowohl als auch AWS IoT Events für die AWS Services auswählen, mit denen Sie arbeiten möchten. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS IoT Events -Endpunkte und -Kontingente in Allgemeine Amazon Web Services-Referenz.
> - Sie müssen dieselbe AWS Region verwenden, wenn Sie andere AWS Ressourcen für die AWS IoT Events Aktionen erstellen. Wenn Sie AWS Regionen wechseln, haben Sie möglicherweise Probleme mit dem Zugriff auf die AWS Ressourcen.

Standardmäßig AWS IoT Events generiert eine Standardnutzlast in JSON für jede Aktion. Diese Aktionsnutzlast enthält alle Attribut-Wert-Paare, die die Informationen über die Detektormodell-

Instance und das Ereignis enthalten, das die Aktion ausgelöst hat. Um die Aktionsnutzlast zu konfigurieren, können Sie einen Inhaltsausdruck verwenden. Weitere Informationen finden Sie unter Ausdrücke und im Datentyp Nutzlast in der AWS IoT Events API-Referenz zu .

AWS IoT Core

IoT topic publish action

Mit der AWS IoT Core Aktion können Sie eine MQTT-Nachricht über den AWS IoT Message Broker veröffentlichen. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS IoT Core -Endpunkte und -Kontingente in Allgemeine Amazon Web Services-Referenz.

Der AWS IoT Message Broker verbindet AWS IoT Clients, indem er Nachrichten von veröffentlichenden Clients an abonnierende Clients sendet. Weitere Informationen finden Sie unter Message Broker für AWS IoT im AWS IoT -Entwicklerhandbuch.

More information (2)

Wenn Sie eine MQTT-Nachricht veröffentlichen, müssen Sie die folgenden Parameter angeben.

`mqttTopic`

Das MQTT-Thema, das die Nachricht empfängt.

Sie können einen MQTT-Themennamen zur Laufzeit dynamisch mithilfe von Variablen oder Eingabewerten definieren, die im Detektormodell erstellt wurden.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die `-iot:Publish`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter IotTopicPublishAction in der AWS IoT Events -API-Referenz.

AWS IoT Events

IoT Events action

Mit der AWS IoT Events Aktion können Sie Daten AWS IoT Events als Eingabe an senden. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS IoT Events -Endpunkte und -Kontingente in Allgemeine Amazon Web Services-Referenz.

AWS IoT Events Mit können Sie Ihre Geräte oder Geräteflotten auf Fehler oder Änderungen im Betrieb überwachen und Aktionen auslösen, wenn solche Ereignisse eintreten. Weitere Informationen finden Sie unter Was ist AWS IoT Events? im AWS IoT Events -Entwicklerhandbuch.

More information (2)

Wenn Sie Daten an senden AWS IoT Events, müssen Sie die folgenden Parameter angeben.

`inputName`

Der Name der AWS IoT Events Eingabe, die die Daten empfängt.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die `-iotevents:BatchPutMessage`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter IotEventsAction in der AWS IoT Events -API-Referenz.

AWS IoT SiteWise

IoT SiteWise action

Mit der AWS IoT SiteWise Aktion können Sie Daten an eine Komponenteneigenschaft in senden AWS IoT SiteWise. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS IoT SiteWise -Endpunkte und -Kontingente in Allgemeine Amazon Web Services-Referenz.

AWS IoT SiteWise ist ein verwalteter Service, mit dem Sie Daten von Industrieanlagen erfassen, organisieren und analysieren können. Weitere Informationen finden Sie unter Was ist AWS IoT SiteWise? im AWS IoT SiteWise -Benutzerhandbuch.

More information (11)

Wenn Sie Daten an eine Asset-Eigenschaft in senden AWS IoT SiteWise, müssen Sie die folgenden Parameter angeben.

> **Important**
>
> Um die Daten zu empfangen, müssen Sie eine vorhandene Komponenteneigenschaft in verwenden AWS IoT SiteWise.
>
> - Wenn Sie die AWS IoT Events Konsole verwenden, müssen Sie angeben, `propertyAlias` um die Zielkomponenteneigenschaft zu identifizieren.
> - Wenn Sie die verwenden AWS CLI, müssen Sie entweder `propertyAlias` oder beides `assetId` und angeben, `propertyId` um die Eigenschaft der Zielkomponente zu identifizieren.
>
> Weitere Informationen finden Sie unter Zuordnung von industriellen Datenströmen zu Komponenten-Eigenschaften im Benutzerhandbuch für AWS IoT SiteWise .

`propertyAlias`

(Optional) Der Alias der Asset-Eigenschaft. Sie können auch einen Ausdruck angeben.

`assetId`

(Optional) Die ID der Komponente, die die angegebene Eigenschaft hat. Sie können auch einen Ausdruck angeben.

propertyId

(Optional) Die ID der Komponenteneigenschaft. Sie können auch einen Ausdruck angeben.

entryId

(Optional) Ein eindeutiger Bezeichner für diesen Eintrag. Sie können die Eintrags-ID verwenden, um zu verfolgen, welcher Dateneintrag im Fehlerfall einen Fehler verursacht. Der Standardwert ist ein neuer eindeutiger Bezeichner. Sie können auch einen Ausdruck angeben.

propertyValue

Eine Struktur, die Details zum Eigenschaftswert enthält.

quality

(Optional) Die Qualität des Asset-Eigenschaftswerts. Der Wert muss `GOOD`, `BAD` oder `UNCERTAIN` lauten. Sie können auch einen Ausdruck angeben.

timestamp

(Optional) Eine Struktur, die Zeitstempelinformationen enthält. Wenn Sie diesen Wert nicht angeben, ist der Standardwert die Ereigniszeit.

timeInSeconds

Der Zeitstempel (in Sekunden) im Unix-Epoch-Format. Der gültige Bereich liegt zwischen 1-31556889864403199. Sie können auch einen Ausdruck angeben.

offsetInNanos

(Optional) Der aus konvertierte Nanosekunden-Offset`timeInSeconds`. Der gültige Bereich liegt zwischen 0-999999999. Sie können auch einen Ausdruck angeben.

value

Eine Struktur, die einen Asset-Eigenschaftswert enthält.

> **Important**
>
> Sie müssen je nach `dataType` der angegebenen Asset-Eigenschaft einen der folgenden Werttypen angeben. Weitere Informationen finden Sie unter AssetProperty in der AWS IoT SiteWise -API-Referenz.

`booleanValue`

(Optional) Der Wert der Asset-Eigenschaft ist ein boolescher Wert, der `TRUE` oder sein muss`FALSE`. Sie können auch einen Ausdruck angeben. Wenn Sie einen Ausdruck verwenden, sollte das ausgewertete Ergebnis ein boolescher Wert sein.

`doubleValue`

(Optional) Der Wert der Asset-Eigenschaft ist ein Double. Sie können auch einen Ausdruck angeben. Wenn Sie einen Ausdruck verwenden, sollte das ausgewertete Ergebnis ein Double sein.

`integerValue`

(Optional) Der Wert der Asset-Eigenschaft ist eine Ganzzahl. Sie können auch einen Ausdruck angeben. Wenn Sie einen Ausdruck verwenden, sollte das ausgewertete Ergebnis eine Ganzzahl sein.

`stringValue`

(Optional) Der Wert der Asset-Eigenschaft ist eine Zeichenfolge. Sie können auch einen Ausdruck angeben. Wenn Sie einen Ausdruck verwenden, sollte das ausgewertete Ergebnis eine Zeichenfolge sein.

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die `-iotsitewise:BatchPutAssetPropertyValue`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter IotSiteWiseAction in der AWS IoT Events -API-Referenz.

Amazon DynamoDB

DynamoDB action

Mit der Amazon-DynamoDB-Aktion können Sie Daten an eine DynamoDB-Tabelle senden. Eine Spalte der DynamoDB-Tabelle empfängt alle Attribut-Wert-Paare in der von Ihnen angegebenen Aktionsnutzlast. Eine Liste der unterstützten Regionen finden Sie unter Amazon-DynamoDB-Endpunkte und -Kontingente im Allgemeine Amazon Web Services-Referenz.

Amazon DynamoDB ist ein vollständig verwalteter NoSQL-Datenbank-Service, der schnelle und planbare Leistung mit nahtloser Skalierbarkeit bereitstellt. Weitere Informationen finden Sie unter Was ist DynamoDB ?im Amazon-DynamoDB-Entwicklerhandbuch.

More information (10)

Wenn Sie Daten an eine Spalte einer DynamoDB-Tabelle senden, müssen Sie die folgenden Parameter angeben.

`tableName`

Der Name der DynamoDB-Tabelle, die die Daten empfängt. Der `tableName` Wert muss mit dem Tabellennamen der DynamoDB-Tabelle übereinstimmen. Sie können auch einen Ausdruck angeben.

`hashKeyField`

Der Name des Hash-Schlüssels (auch Partitionsschlüssel genannt). Der `hashKeyField` Wert muss mit dem Partitionsschlüssel der DynamoDB-Tabelle übereinstimmen. Sie können auch einen Ausdruck angeben.

`hashKeyType`

(Optional) Der Datentyp des Hash-Schlüssels. Der Wert des Hash-Schlüsseltyps muss `STRING` oder sein`NUMBER`. Der Standardwert ist `STRING`. Sie können auch einen Ausdruck angeben.

`hashKeyValue`

Der Wert des Hash-Schlüssels Die `hashKeyValue` verwendet Ersatzvorlagen. Diese Vorlagen stellen Daten zur Laufzeit bereit. Sie können auch einen Ausdruck angeben.

`rangeKeyField`

(Optional) Der Name des Bereichsschlüssels (auch Sortierschlüssel genannt). Der `rangeKeyField` Wert muss mit dem Sortierschlüssel der DynamoDB-Tabelle übereinstimmen. Sie können auch einen Ausdruck angeben.

`rangeKeyType`

(Optional) Der Datentyp des Bereichsschlüssels. Der Wert des Hash-Schlüsseltyps muss `STRING` oder sein`NUMBER`. Der Standardwert ist `STRING`. Sie können auch einen Ausdruck angeben.

`rangeKeyValue`

(Optional) Der Wert des Bereichsschlüssels. Die `rangeKeyValue` verwendet Ersatzvorlagen. Diese Vorlagen stellen Daten zur Laufzeit bereit. Sie können auch einen Ausdruck angeben.

operation

(Optional) Der Typ der auszuführenden Operation. Sie können auch einen Ausdruck angeben. Der Operationswert muss einer der folgenden Werte sein:

- `INSERT` – Fügt Daten als neues Element in die DynamoDB-Tabelle ein. Dies ist der Standardwert.
- `UPDATE` – Aktualisiert ein vorhandenes Element der DynamoDB-Tabelle mit neuen Daten.
- `DELETE` – Löschen Sie ein vorhandenes Element aus der DynamoDB-Tabelle.

`payloadField`

(Optional) Der Name der DynamoDB-Spalte, die die Aktionsnutzlast empfängt. Der Standardname lautet `payload`. Sie können auch einen Ausdruck angeben.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

Wenn der angegebene Nutzlasttyp eine Zeichenfolge ist, `DynamoDBAction` sendet Nicht-JSON-Daten als Binärdaten an die DynamoDB-Tabelle. Die DynamoDB-Konsole zeigt die Daten als Base64-codierten Text an. Der Wert von `payloadField` ist *`payload-field`*`_raw`. Sie können auch einen Ausdruck angeben.

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die -`dynamodb:PutItem`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter DynamoDBAction in der APIAWS IoT Events -Referenz zu .

Amazon DynamoDB (v2)

DynamoDBv2 action

Mit der Amazon DynamoDB (v2)-Aktion können Sie Daten in eine DynamoDB-Tabelle schreiben. Eine separate Spalte der DynamoDB-Tabelle erhält ein Attribut-Wert-Paar in der von Ihnen angegebenen Aktionsnutzlast. Eine Liste der unterstützten Regionen finden Sie unter Amazon-DynamoDB-Endpunkte und -Kontingente im Allgemeine Amazon Web Services-Referenz.

Amazon DynamoDB ist ein vollständig verwalteter NoSQL-Datenbank-Service, der schnelle und planbare Leistung mit nahtloser Skalierbarkeit bereitstellt. Weitere Informationen finden Sie unter Was ist DynamoDB ?im Amazon-DynamoDB-Entwicklerhandbuch.

More information (2)

Wenn Sie Daten an mehrere Spalten einer DynamoDB-Tabelle senden, müssen Sie die folgenden Parameter angeben.

`tableName`

Der Name der DynamoDB-Tabelle, die die Daten empfängt. Sie können auch einen Ausdruck angeben.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Important**
>
> Der Nutzlasttyp muss JSON sein. Sie können auch einen Ausdruck angeben.

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die -`dynamodb:PutItem`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter DynamoDBv2Action in der APIAWS IoT Events -Referenz zu .

Amazon Data Firehose

Firehose action

Mit der Amazon-Data-Firehose-Aktion können Sie Daten an einen Firehose-Bereitstellungsdatenstrom senden. Eine Liste der unterstützten Regionen finden Sie unter Endpunkte und Kontingente von Amazon Data Firehose im Allgemeine Amazon Web Services-Referenz.

Amazon Data Firehose ist ein vollständig verwalteter Service für die Bereitstellung von Echtzeit-Streaming-Daten an Ziele wie Amazon Simple Storage Service (Amazon Simple Storage Service), Amazon Redshift, Amazon OpenSearch Service (OpenSearch Service) und Splunk. Weitere Informationen finden Sie unter Was ist Amazon Data Firehose? im Entwicklerhandbuch für Amazon Data Firehose.

More information (3)

Wenn Sie Daten an einen Firehose-Bereitstellungs-Stream senden, müssen Sie die folgenden Parameter angeben.

`deliveryStreamName`

Der Name des Firehose-Bereitstellungs-Streams, der die Daten empfängt.

`separator`

(Optional) Sie können ein Zeichentrennzeichen verwenden, um kontinuierliche Daten zu trennen, die an den Firehose-Bereitstellungsdatenstrom gesendet werden. Der Trennzeichenwert muss `'\n'`(neue Zeile), `'\t'` (Registerkarte), `'\r\n'` (neue Windows-Zeile) oder `','` (Komma) sein.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die `-firehose:PutRecord`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter FirehoseAction in der AWS IoT Events -API-Referenz.

AWS Lambda

Lambda action

Mit der AWS Lambda Aktion können Sie eine Lambda-Funktion aufrufen. Eine vollständige Liste der unterstützten Regionen finden Sie unter AWS Lambda -Endpunkte und -Kontingente in Allgemeine Amazon Web Services-Referenz.

AWS Lambda ist ein Datenverarbeitungsservice, mit dem Sie Code ausführen können, ohne Server bereitstellen oder verwalten zu müssen. Weitere Informationen finden Sie unter Was ist AWS Lambda? im AWS Lambda -Entwicklerhandbuch.

More information (2)

Wenn Sie eine Lambda-Funktion aufrufen, müssen Sie die folgenden Parameter angeben.

`functionArn`

Der ARN der aufzurufenden Lambda-Funktion.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Note**
>
> Stellen Sie sicher, dass die Ihrer AWS IoT Events Servicerolle zugeordnete Richtlinie die `-lambda:InvokeFunction`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter LambdaAction in der AWS IoT Events -API-Referenz.

Amazon Simple Notification Service

SNS action

Mit der Aktion zum Veröffentlichen von Amazon SNS-Themen können Sie eine Amazon SNSNachricht veröffentlichen. Eine Liste der unterstützten Regionen finden Sie unter Endpunkte und Kontingente von Amazon Simple Notification Service im Allgemeine Amazon Web Services-Referenz.

Amazon Simple Notification Service (Amazon Simple Notification Service) ist ein Webservice, der die Zustellung oder das Senden von Nachrichten an abonnierende Endpunkte oder Clients koordiniert und verwaltet. Weitere Informationen finden Sie unter Was ist Amazon SNS ?im Amazon Simple Notification Service-Entwicklerhandbuch.

> **Note**
>
> Die Amazon SNS-Themenveröffentlichungsaktion unterstützt keine Amazon SNS-FIFO-Themen (First in, First out). Da es sich bei der Regel-Engine um einen vollständig verteilten Service handelt, werden die Nachrichten möglicherweise nicht in einer bestimmten Reihenfolge angezeigt, wenn die Amazon SNS-Aktion initiiert wird.

More information (2)

Wenn Sie eine Amazon SNS-Nachricht veröffentlichen, müssen Sie die folgenden Parameter angeben.

`targetArn`

Der ARN des Amazon SNS-Ziels, das die Nachricht empfängt.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die `-sns:Publish`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter SNSTopicPublishAction in der APIAWS IoT Events -Referenz zu .

Amazon Simple Queue Service

SQS action

Mit der Amazon SQS-Aktion können Sie Daten an eine Amazon SQSWarteschlange senden. Eine Liste der unterstützten Regionen finden Sie unter Endpunkte und Kontingente von Amazon Simple Queue Service im Allgemeine Amazon Web Services-Referenz.

Amazon Simple Queue Service (Amazon SQS) bietet eine sichere, dauerhafte und verfügbare gehostete Warteschlange, die es Ihnen ermöglicht, verteilte Softwaresysteme und -komponenten zu integrieren und zu entkoppeln. Weitere Informationen finden Sie unter Was ist Amazon Simple Queue Service>? im Amazon Simple Queue Service-Entwicklerhandbuch.

> **Note**
>
> Die Amazon SQS-Aktion unterstützt keine Amazon SQS-FIFO-Themen (First in, First out). Da es sich bei der Regel-Engine um einen vollständig verteilten Service handelt, werden die Nachrichten möglicherweise nicht in einer bestimmten Reihenfolge angezeigt, wenn die Amazon SQS-Aktion initiiert wird.

More information (3)

Wenn Sie Daten an eine Amazon SQS-Warteschlange senden, müssen Sie die folgenden Parameter angeben.

`queueUrl`

Die URL der Amazon SQS-Warteschlange, die die Daten empfängt.

`useBase64`

(Optional) AWS IoT Events kodiert die Daten in Base64-Text, wenn Sie angeben`TRUE`. Der Standardwert ist `FALSE`.

`payload`

(Optional) Die Standardnutzlast enthält alle Attribut-Wert-Paare, die über die Informationen über die Detektormodell-Instance und das Ereignis verfügen, das die Aktion ausgelöst hat. Sie können auch die Nutzlast anpassen. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events -Referenz zu .

> **Note**
>
> Stellen Sie sicher, dass die mit Ihrer AWS IoT Events Servicerolle verknüpfte Richtlinie die -`sqs:SendMessage`Berechtigung erteilt. Weitere Informationen finden Sie unter Identity and Access Management für AWS IoT Events.

Weitere Informationen finden Sie unter SNSTopicPublishAction in der APIAWS IoT Events -Referenz zu .

Sie können auch Amazon SNS und die AWS IoT Core Regel-Engine verwenden, um eine AWS Lambda -Funktion auszulösen. Dadurch ist es möglich, Aktionen mit anderen -Services wie Amazon Connect oder sogar einer Enterprise Resource Planning (ERP)-Anwendung zu ergreifen.

> **Note**
>
> Um große Streams von Datensätzen in Echtzeit zu erfassen und zu verarbeiten, können Sie andere - AWS Services wie Amazon Kinesis verwenden. Von dort aus können Sie eine

erste Analyse abschließen und die Ergebnisse dann AWS IoT Events als Eingabe an einen Detektor senden.

Ausdrücke

AWS IoT Eventsbietet mehrere Möglichkeiten, Werte anzugeben, wenn Sie Detektormodelle erstellen und aktualisieren. Sie können Ausdrücke verwenden, um Literalwerte anzugeben, oder Sie AWS IoT Events können die Ausdrücke auswerten, bevor Sie bestimmte Werte angeben.

Syntax

Sie können Literale, Operatoren, Funktionen, Verweise und Ersetzungsvorlagen in den AWS IoT Events Ausdrücken verwenden.

Literale

- Ganzzahl
- Dezimal
- Zeichenfolge
- Boolesch

Operatoren

Unär

- Nicht (Boolean): `!`
- Nicht (bitweise): `~`
- Minus (arithmetisch): `-`

Zeichenfolge

- Verkettung: **+**

  Beide Operanden müssen Zeichenketten sein. Zeichenkettenliterale müssen in einfache Anführungszeichen (') eingeschlossen werden.

  Zum Beispiel: -> `'my' + 'string' 'mystring'`

Arithmetisch

- Zusatz: **+**

Beide Operanden müssen numerisch sein.

- Subtraktion: **-**
- Einteilung: **/**

  Das Ergebnis der Division ist ein gerundeter Ganzzahlwert, sofern nicht mindestens einer der Operanden (Divisor oder Dividend) ein Dezimalwert ist.
- Multiplikation: *****

Bitweise (Ganzzahl)

- ODER: **|**

  Zum Beispiel: `13 | 5 -> 13`
- UND: **&**

  Zum Beispiel: `13 & 5 -> 5`
- XOR: **^**

  Zum Beispiel: `-> 13 ^ 5 8`
- NICHT: **~**

  Zum Beispiel: `~13 -> -14`

Boolesch

- Weniger als: **<**
- Weniger als oder gleich: **<=**
- Gleich: **==**
- Nicht gleich: **!=**
- Größer als oder gleich: **>=**
- Größer als: **>**
- UND: **&&**
- ODER: **||**

> **Note**
>
> Wenn ein Unterausdruck von undefinierte Daten `||` enthält, wird dieser Unterausdruck als behandelt. `false`

Klammern

Sie können Klammern verwenden, um Begriffe innerhalb eines Ausdrucks zu gruppieren.

Funktionen

Integrierte Funktionen

`timeout``("`*`timer-name`*`")`

Prüft, `true` ob der angegebene Timer abgelaufen ist. Ersetzen Sie "*`Timer-Name`*" durch den Namen eines Timers, den Sie definiert haben, in Anführungszeichen. In einer Ereignisaktion können Sie einen Timer definieren und dann den Timer starten, zurücksetzen oder einen zuvor definierten Timer löschen. Sehen Sie sich das Feld an`detectorModelDefinition.states.onInput|onEnter|onExit.events.actions.setTimer.timerName`.

Auf einen Timer, der in einem Status festgelegt ist, kann in einem anderen Status verwiesen werden. Sie müssen den Status besuchen, in dem Sie den Timer erstellt haben, bevor Sie den Status aufrufen, in dem auf den Timer verwiesen wird.

Ein Detektormodell hat beispielsweise zwei Zustände, TemperatureChecked und RecordUpdated. Sie haben im TemperatureChecked Bundesstaat einen Timer erstellt. Sie müssen den TemperatureChecked Bundesstaat zuerst besuchen, bevor Sie den Timer im RecordUpdated Bundesstaat verwenden können.

Um die Genauigkeit zu gewährleisten, sollte die Mindestzeit, für die ein Timer eingestellt werden muss, 60 Sekunden betragen.

> **Note**
>
> `timeout()`gibt `true` nur das erste Mal zurück, wenn der Timer nach Ablauf des Timers zum ersten Mal überprüft wird, und kehrt `false` danach zurück.

`convert``(`*`type`*`, `*`expression`*`)`

Ergibt den Wert des Ausdrucks, der in den angegebenen Typ konvertiert wurde. Der *`Typwert`* muss `String``Boolean`, oder `Decimal` sein. Verwenden Sie eines dieser Schlüsselwörter

oder einen Ausdruck, der eine Zeichenfolge ergibt, die das Schlüsselwort enthält. Nur die folgenden Konvertierungen sind erfolgreich und geben einen gültigen Wert zurück:

- Boolean -> Zeichenfolge

  Gibt die Zeichenfolge `"true"` oder zurück. `"false"`
- Dezimal -> Zeichenfolge
- Zeichenfolge -> Boolean
- Zeichenfolge -> Dezimal

  Die angegebene Zeichenfolge muss eine gültige Darstellung einer Dezimalzahl sein, andernfalls schlägt sie `convert()` fehl.

Wenn `convert()` kein gültiger Wert zurückgegeben wird, ist der Ausdruck, zu dem er gehört, ebenfalls ungültig. Dieses Ergebnis entspricht dem Ereignis, in dem der `actions` Ausdruck auftritt, `false` und löst den Übergang zum `nextState` angegebenen Wert nicht aus.

`isNull`(*`expression`*)

Wird ausgewertet, `true` ob der Ausdruck Null zurückgibt. Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "a": null }`, wird Folgendes als ausgewertet`true`, aber als `isUndefined($input.MyInput.a)` ausgewertet. `false`

```
isNull($input.MyInput.a)
```

`isUndefined`(*`expression`*)

Wird als undefiniert ausgewertet, `true` wenn der Ausdruck undefiniert ist. Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "a": null }`, wird Folgendes als ausgewertet`false`, aber `isNull($input.MyInput.a)` als ausgewertet. `true`

```
isUndefined($input.MyInput.a)
```

`triggerType`("*`type`*")

Der *`Typwert`* kann oder sein. `"Message"` `"Timer"` Prüft, `true` ob die Ereignisbedingung, in der sie auftritt, ausgewertet wird, weil ein Timer abgelaufen ist, wie im folgenden Beispiel.

```
triggerType("Timer")
```

Oder es wurde eine Eingabenachricht empfangen.

```
triggerType("Message")
```

currentInput("*input*")

Prüft, `true` ob die Ereignisbedingung, in der sie auftritt, ausgewertet wird, weil die angegebene Eingabemeldung empfangen wurde. Wenn die Eingabe beispielsweise die Nachricht `Command` empfängt`{ "value": "Abort" }`, wird Folgendes als ausgewertet. `true`

```
currentInput("Command")
```

Verwenden Sie diese Funktion, um zu überprüfen, ob die Bedingung ausgewertet wird, weil eine bestimmte Eingabe empfangen wurde und ein Timer nicht abgelaufen ist, wie im folgenden Ausdruck dargestellt.

```
currentInput("Command") && $input.Command.value == "Abort"
```

Funktionen zum Abgleich von Zeichenketten

startsWith(*expression1*, *expression2*)

Prüft, `true` ob der erste Zeichenkettenausdruck mit dem zweiten Zeichenkettenausdruck beginnt. Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "status": "offline"}`, wird Folgendes als ausgewertet. `true`

```
startsWith($input.MyInput.status, "off")
```

Beide Ausdrücke müssen einen Zeichenkettenwert ergeben. Wenn einer der Ausdrücke keinen Zeichenkettenwert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

endsWith(*expression1*, *expression2*)

Prüft, `true` ob der erste Zeichenkettenausdruck mit dem zweiten Zeichenkettenausdruck endet. Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "status": "offline" }`, wird Folgendes als ausgewertet. `true`

```
endsWith($input.MyInput.status, "line")
```

Beide Ausdrücke müssen einen Zeichenkettenwert ergeben. Wenn einer der Ausdrücke keinen Zeichenkettenwert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

`contains`(*`expression1`*, *`expression2`*)

Prüft, `true` ob der erste Zeichenkettenausdruck den zweiten Zeichenkettenausdruck enthält. Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "status": "offline" }`, wird Folgendes als ausgewertet. `true`

```
contains($input.MyInput.value, "fli")
```

Beide Ausdrücke müssen einen Zeichenkettenwert ergeben. Wenn einer der Ausdrücke keinen Zeichenkettenwert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

Funktionen zur bitweisen Ganzzahlmanipulation

`bitor`(*`expression1`*, *`expression2`*)

Wertet das bitweise ODER der Integer-Ausdrücke aus (die binäre OR-Operation wird für die entsprechenden Bits der Ganzzahlen ausgeführt). Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "value1": 13, "value2": 5 }`, wird Folgendes als ausgewertet. `13`

```
bitor($input.MyInput.value1, $input.MyInput.value2)
```

Beide Ausdrücke müssen einen ganzzahligen Wert ergeben. Wenn einer der Ausdrücke keinen ganzzahligen Wert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

`bitand`(*`expression1`*, *`expression2`*)

Wertet das bitweise UND der Integer-Ausdrücke aus (die binäre AND-Operation wird für die entsprechenden Bits der Ganzzahlen ausgeführt). Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "value1": 13, "value2": 5 }`, wird Folgendes als ausgewertet. `5`

```
bitand($input.MyInput.value1, $input.MyInput.value2)
```

Beide Ausdrücke müssen einen ganzzahligen Wert ergeben. Wenn einer der Ausdrücke keinen ganzzahligen Wert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

`bitxor``(`*`expression1`*`, `*`expression2`*`)`

Wertet das bitweise XOR der Integer-Ausdrücke aus (die binäre XOR-Operation wird für die entsprechenden Bits der Ganzzahlen ausgeführt). Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "value1": 13, "value2": 5 }`, wird Folgendes als ausgewertet. 8

```
bitxor($input.MyInput.value1, $input.MyInput.value2)
```

Beide Ausdrücke müssen einen ganzzahligen Wert ergeben. Wenn einer der Ausdrücke keinen ganzzahligen Wert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

`bitnot``(`*`expression`*`)`

Wertet das bitweise NOT des Integer-Ausdrucks aus (die binäre NOT-Operation wird für die Bits der Ganzzahl ausgeführt). Wenn die Eingabe beispielsweise die Nachricht `MyInput` empfängt`{ "value": 13 }`, wird Folgendes als ausgewertet. `-14`

```
bitnot($input.MyInput.value)
```

Beide Ausdrücke müssen einen ganzzahligen Wert ergeben. Wenn einer der Ausdrücke keinen ganzzahligen Wert ergibt, ist das Ergebnis der Funktion undefiniert. Es werden keine Konvertierungen durchgeführt.

Referenzen

Eingaben

`$input.`*`input-name`*`.`*`path-to-data`*

`input-name`ist eine Eingabe, die Sie mithilfe der CreateInputAktion erstellen.

Wenn Sie beispielsweise eine Eingabe benannt haben, `TemperatureInput` für die Sie `inputDefinition.attributes.jsonPath` Einträge definiert haben, werden die Werte möglicherweise in den folgenden verfügbaren Feldern angezeigt.

```
{
    "temperature": 78.5,
    "date": "2018-10-03T16:09:09Z"
  }
```

Verwenden Sie den folgenden Befehl, um `temperature` auf den Wert des Felds zu verweisen.

```
$input.TemperatureInput.temperature
```

Bei Feldern, deren Werte Arrays sind, können Sie mit Hilfe `[`*n*`]` von auf Elemente des Arrays verweisen. Nehmen wir zum Beispiel die folgenden Werte an:

```
{
    "temperatures": [
      78.4,
      77.9,
      78.8
    ],
    "date": "2018-10-03T16:09:09Z"
  }
```

Auf den Wert `78.8` kann mit dem folgenden Befehl verwiesen werden.

```
$input.TemperatureInput.temperatures[2]
```

Variablen

`$variable.`*`variable-name`*

Das *`variable-name`* ist eine Variable, die Sie mithilfe der CreateDetectorModelAktion definiert haben.

Wenn Sie beispielsweise eine Variable benannt haben, `TechnicianID` die Sie mithilfe definiert habendetectorDefinition.states.onInputEvents.actions.setVariable.variableName, können Sie mit dem folgenden Befehl auf den Wert (Zeichenfolge) verweisen, der der Variablen zuletzt zugewiesen wurde.

```
$variable.TechnicianID
```

Sie können die Werte von Variablen nur mithilfe der `setVariable` Aktion festlegen. Sie können Variablen in einem Ausdruck keine Werte zuweisen. Eine Variable kann nicht rückgängig gemacht werden. Sie können ihr beispielsweise den Wert `null` nicht zuweisen.

> **Note**
>
> In Verweisen, die Bezeichner verwenden, die nicht dem Muster (regulärer Ausdruck) folgen`[a-zA-Z][a-zA-Z0-9_]*`, müssen Sie diese Bezeichner in Backticks () einschließen. ` Beispielsweise `_value` muss ein Verweis auf eine Eingabe, die `MyInput` mit einem Feld benannt ist, dieses Feld als angeben. `$input.MyInput.`_value``

Wenn Sie Verweise in Ausdrücken verwenden, überprüfen Sie Folgendes:

- Wenn Sie eine Referenz als Operanden mit einem oder mehreren Operatoren verwenden, stellen Sie sicher, dass alle Datentypen, auf die Sie verweisen, kompatibel sind.

  Im folgenden Ausdruck 2 ist Integer beispielsweise ein Operand sowohl der == Operatoren als auch. && Um sicherzustellen, dass die Operanden kompatibel sind `$variable.testVariable + 1` und auf eine Ganzzahl oder Dezimalzahl verweisen `$variable.testVariable` müssen.

  Außerdem 1 ist Integer ein Operand des Operators+. `$variable.testVariable`Muss daher auf eine Ganzzahl oder Dezimalzahl verweisen.

  ```
  ‘$variable.testVariable + 1 == 2 && $variable.testVariable’
  ```

- Wenn Sie eine Referenz als Argument verwenden, das an eine Funktion übergeben wird, stellen Sie sicher, dass die Funktion die Datentypen unterstützt, auf die Sie verweisen.

  Für die folgende `timeout("`*`time-name`*`")` Funktion ist beispielsweise eine Zeichenfolge mit doppelten Anführungszeichen als Argument erforderlich. Wenn Sie eine Referenz für den *`Timer-Name-Wert`* verwenden, müssen Sie auf eine Zeichenfolge mit doppelten Anführungszeichen verweisen.

  ```
  timeout("timer-name")
  ```

> **Note**
>
> Wenn Sie für die `convert(type, expression)` Funktion eine Referenz für den *`Typwert`* verwenden, muss das ausgewertete Ergebnis Ihrer Referenz`String`,`Decimal`, oder sein. `Boolean`

AWS IoT EventsAusdrücke unterstützen die Datentypen Integer, Decimal, String und Boolean. Die folgende Tabelle enthält eine Liste inkompatibler Typpaare.

Inkompatible Typenpaare
Ganzzahl, Zeichenfolge
Ganzzahl, Boolean
Dezimal, Zeichenfolge
Dezimal, Boolesch
Zeichenfolge, Boolean

Ersetzungsvorlagen

`'${`*`expression`*`}'`

Der `${}` identifiziert die Zeichenfolge als interpolierte Zeichenfolge. Das `expression` kann ein beliebiger Ausdruck sein. AWS IoT Events Dazu gehören Operatoren, Funktionen und Verweise.

Sie haben die SetVariableActionAktion beispielsweise verwendet, um eine Variable zu definieren. `variableName` ist `SensorID` und `value` ist `10`. Sie können die folgenden Substitutionsvorlagen erstellen.

Substitutionsvorlage	Ergebniszeichenfolge
`'${'Sensor ' + $variable.SensorID}'`	`"Sensor 10"`

Substitutionsvorlage	Ergebniszeichenfolge
`'Sensor ' + '${$variable.SensorID + 1}'`	`"Sensor 11"`
`'Sensor 10: ${$variable.SensorID == 10}'`	`"Sensor 10: true"`
`'{\"sensor\":\"${$variable.SensorID + 1}\"}'`	`"{\"sensor\":\"11\"}"`
`'{\"sensor\":${$variable.SensorID + 1}}'`	`"{\"sensor\":11}"`

Verwendung von Ausdrücken

Sie können Werte in einem Detektormodell auf folgende Weise angeben:

- Geben Sie unterstützte Ausdrücke in der AWS IoT Events Konsole ein.
- Übergeben Sie die Ausdrücke als Parameter an die AWS IoT Events APIs.

Ausdrücke unterstützen Literale, Operatoren, Funktionen, Referenzen und Ersatzvorlagen.

> **Important**
>
> Ihre Ausdrücke müssen auf eine Ganzzahl, eine Dezimalzahl, eine Zeichenfolge oder einen booleschen Wert verweisen.

Ausdrücke schreiben AWS IoT Events

Sehen Sie sich die folgenden Beispiele an, die Ihnen beim Schreiben Ihrer AWS IoT Events Ausdrücke helfen sollen:

Literal

Bei Literalwerten müssen die Ausdrücke einfache Anführungszeichen enthalten. Ein boolescher Wert muss entweder oder `true` sein. `false`

```
'123'        # Integer
'123.12'     # Decimal
'hello'      # String
'true'       # Boolean
```

Referenz

Bei Referenzen müssen Sie entweder Variablen oder Eingabewerte angeben.

- Die folgende Eingabe bezieht sich auf eine Dezimalzahl,. `10.01`

  ```
  $input.GreenhouseInput.temperature
  ```

- Die folgende Variable verweist auf eine Zeichenfolge,`Greenhouse Temperature Table`.

  ```
  $variable.TableName
  ```

Vorlage für die Substitution

Für eine Substitutionsvorlage müssen Sie `${}` verwenden und die Vorlage muss von einfachen Anführungszeichen umschlossen sein. Eine Substitutionsvorlage kann auch eine Kombination aus Literalen, Operatoren, Funktionen, Referenzen und Substitutionsvorlagen enthalten.

- Das ausgewertete Ergebnis des folgenden Ausdrucks ist eine Zeichenfolge,`50.018 in Fahrenheit`.

  ```
  '${$input.GreenhouseInput.temperature * 9 / 5 + 32} in Fahrenheit'
  ```

- Das ausgewertete Ergebnis des folgenden Ausdrucks ist eine Zeichenfolge,`{\"sensor_id\":\"Sensor_1\",\"temperature\":\"50.018\"}`.

  ```
  '{\"sensor_id\":\"${$input.GreenhouseInput.sensors[0].sensor1}\",\"temperature\":
  \"${$input.GreenhouseInput.temperature*9/5+32}\"}'
  ```

Zeichenfolgenverkettung

Für eine Zeichenfolgeverkettung müssen Sie + verwenden. Eine Zeichenfolgeverkettung kann auch eine Kombination aus Literalen, Operatoren, Funktionen, Referenzen und Substitutionsvorlagen enthalten.

- Das ausgewertete Ergebnis des folgenden Ausdrucks ist eine Zeichenfolge,`Greenhouse Temperature Table 2000-01-01`.

```
'Greenhouse Temperature Table ' + $input.GreenhouseInput.date
```

Beispiele für Detektormodelle

Dieser Abschnitt enthält Beispiele für Detektormodelle und Eingänge.

Themen

HVAC-Temperatursteuerung

Hintergrundgeschichte

In diesem Beispiel wird ein Temperaturregelungsmodell (ein Thermostat) mit folgenden Funktionen implementiert:

- Ein von Ihnen definiertes Meldermodell, das mehrere Bereiche überwachen und steuern kann. (Für jeden Bereich wird eine Melderinstanz erstellt.)
- Jede Melderinstanz empfängt Temperaturdaten von mehreren Sensoren, die sich in jedem Kontrollbereich befinden.
- Sie können die gewünschte Temperatur (den Sollwert) für jeden Bereich jederzeit ändern.
- Sie können die Betriebsparameter für jeden Bereich definieren und diese Parameter jederzeit ändern.
- Sie können jederzeit Sensoren zu einem Bereich hinzufügen oder Sensoren aus einem Bereich löschen.
- Sie können Heiz- und Kühlgeräte mit einer Mindestlaufzeit aktivieren, um sie vor Beschädigungen zu schützen.
- Die Melder weisen anomale Sensorwerte zurück und melden sie.
- Sie können Sollwerte für die Notfalltemperatur definieren. Wenn ein Sensor eine Temperatur über oder unter den von Ihnen definierten Sollwerten meldet, werden die Heiz- oder Kühlgeräte sofort eingeschaltet und der Melder meldet diese Temperaturspitze.

Dieses Beispiel demonstriert die folgenden Funktionsmöglichkeiten:

- Erstellen Sie Modelle für Ereignisdetektoren.
- Erstellen Sie Eingaben.
- Investieren Sie Eingaben in ein Detektormodell.
- Evaluieren Sie die Triggerbedingungen.
- Beziehen Sie sich auf Zustandsvariablen in Bedingungen und legen Sie die Werte der Variablen abhängig von den Bedingungen fest.
- Beziehen Sie sich auf Timer in Bedingungen und stellen Sie Timer je nach Bedingungen ein.
- Ergreifen Sie Maßnahmen, die Amazon SNS- und MQTT-Nachrichten senden.

Eingabedefinitionen

A "`seedTemperatureInput`" wird verwendet, um eine Melderinstanz für einen Bereich zu erstellen und dessen Betriebsparameter zu definieren.

Verwendeter CLI-Befehl:

```
aws iotevents create-input --cli-input-json file://seedInput.json
```

Datei: `seedInput.json`

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "areaId" },
      { "jsonPath": "desiredTemperature" },
      { "jsonPath": "allowedError" },
      { "jsonPath": "rangeHigh" },
      { "jsonPath": "rangeLow" },
      { "jsonPath": "anomalousHigh" },
      { "jsonPath": "anomalousLow" },
      { "jsonPath": "sensorCount" },
      { "jsonPath": "noDelay" }
    ]
  }
```

```
}
```

Antwort:

```
{
    "inputConfiguration": {
        "status": "ACTIVE",
        "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/
seedTemperatureInput",
        "lastUpdateTime": 1557519620.736,
        "creationTime": 1557519620.736,
        "inputName": "seedTemperatureInput",
        "inputDescription": "Temperature seed values."
    }
}
```

Bei Bedarf "`temperatureInput`" sollte von jedem Sensor in jedem Bereich A gesendet werden.

Verwendeter CLI-Befehl:

```
aws iotevents create-input --cli-input-json file://temperatureInput.json
```

Datei: `temperatureInput.json`

```
{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorId" },
      { "jsonPath": "areaId" },
      { "jsonPath": "sensorData.temperature" }
    ]
  }
}
```

Antwort:

```
{
    "inputConfiguration": {
        "status": "ACTIVE",
```

```
        "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/temperatureInput",
        "lastUpdateTime": 1557519707.399,
        "creationTime": 1557519707.399,
        "inputName": "temperatureInput",
        "inputDescription": "Temperature sensor unit data."
    }
}
```

Definition des Detektormodells

Die "areaDetectorModel" definiert, wie jede Detektorinstanz funktioniert. Jede "state machine" Instanz nimmt Temperatursensormesswerte auf, ändert dann den Status und sendet abhängig von diesen Messwerten Steuermeldungen.

Verwendeter CLI-Befehl:

```
aws iotevents create-detector-model --cli-input-json file://areaDetectorModel.json
```

Datei: areaDetectorModel.json

```
{
  "detectorModelName": "areaDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "start",
        "onEnter": {
          "events": [
            {
              "eventName": "prepare",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "sensorId",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "reportedTemperature",
```

```
                    "value": "0.1"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "resetMe",
                    "value": "false"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "initialize",
              "condition": "$input.seedTemperatureInput.sensorCount > 0",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "rangeHigh",
                    "value": "$input.seedTemperatureInput.rangeHigh"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "rangeLow",
                    "value": "$input.seedTemperatureInput.rangeLow"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "desiredTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "averageTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                },
                {
```

```
                "setVariable": {
                  "variableName": "allowedError",
                  "value": "$input.seedTemperatureInput.allowedError"
                }
              },
              {
                "setVariable": {
                  "variableName": "anomalousHigh",
                  "value": "$input.seedTemperatureInput.anomalousHigh"
                }
              },
              {
                "setVariable": {
                  "variableName": "anomalousLow",
                  "value": "$input.seedTemperatureInput.anomalousLow"
                }
              },
              {
                "setVariable": {
                  "variableName": "sensorCount",
                  "value": "$input.seedTemperatureInput.sensorCount"
                }
              },
              {
                "setVariable": {
                  "variableName": "noDelay",
                  "value": "$input.seedTemperatureInput.noDelay == true"
                }
              }
            ],
            "nextState": "idle"
          },
          {
            "eventName": "reset",
            "condition": "($variable.resetMe == true) &&
($input.temperatureInput.sensorData.temperature < $variable.anomalousHigh &&
$input.temperatureInput.sensorData.temperature > $variable.anomalousLow)",
            "actions": [
              {
                "setVariable": {
                  "variableName": "averageTemperature",
                  "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
```

```
            }
          ],
          "nextState": "idle"
        }
      ]
    },
    "onExit": {
      "events": [
        {
          "eventName": "resetHeatCool",
          "condition": "true",
          "actions": [
            {
              "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
              }
            },
            {
              "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
              }
            },
            {
              "iotTopicPublish": {
                "mqttTopic": "hvac/Heating/Off"
              }
            },
            {
              "iotTopicPublish": {
                "mqttTopic": "hvac/Cooling/Off"
              }
            }
          ]
        }
      ]
    }
  },


  {
    "stateName": "idle",
    "onInput": {
      "events": [
        {
```

```
            "eventName": "whatWasInput",
            "condition": "true",
            "actions": [
              {
                "setVariable": {
                  "variableName": "sensorId",
                  "value": "$input.temperatureInput.sensorId"
                }
              },
              {
                "setVariable": {
                  "variableName": "reportedTemperature",
                  "value": "$input.temperatureInput.sensorData.temperature"
                }
              }
            ]
          },
          {
            "eventName": "changeDesired",
            "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
            "actions": [
              {
                "setVariable": {
                  "variableName": "desiredTemperature",
                  "value": "$input.seedTemperatureInput.desiredTemperature"
                }
              }
            ]
          },
          {
            "eventName": "calculateAverage",
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
            "actions": [
              {
                "setVariable": {
                  "variableName": "averageTemperature",
                  "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
              }
            ]
```

```
            }
          ],
          "transitionEvents": [
            {
              "eventName": "anomalousInputArrived",
              "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/anomaly"
                  }
                }
              ],
              "nextState": "idle"
            },

            {
              "eventName": "highTemperatureSpike",
              "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                  }
                },
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/On"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
```

```
            ],
            "nextState": "cooling"
          },

          {
            "eventName": "lowTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/spike"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                }
              },
              {
                "iotTopicPublish": {
                  "mqttTopic": "hvac/Heating/On"
                }
              },
              {
                "setVariable": {
                  "variableName": "enteringNewState",
                  "value": "true"
                }
              }
            ],
            "nextState": "heating"
          },

          {
            "eventName": "highTemperatureThreshold",
            "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >
($variable.desiredTemperature + $variable.allowedError))",
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
                }
```

```
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/On"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "cooling"
            },

            {
              "eventName": "lowTemperatureThreshold",
              "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <
($variable.desiredTemperature - $variable.allowedError))",
              "actions": [
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "heating"
            }
          ]
        }
      },
```

```
{
  "stateName": "cooling",
  "onEnter": {
    "events": [
      {
        "eventName": "delay",
        "condition": "!$variable.noDelay && $variable.enteringNewState",
        "actions": [
          {
            "setTimer": {
              "timerName": "coolingTimer",
              "seconds": 180
            }
          },
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "false"
            }
          }
        ]
      },
      {
        "eventName": "dontDelay",
        "condition": "$variable.noDelay == true",
        "actions": [
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "true"
            }
          }
        ]
      },
      {
        "eventName": "beenHere",
        "condition": "true",
        "actions": [
          {
            "setVariable": {
              "variableName": "enteringNewState",
              "value": "false"
```

```
                }
              }
            ]
          }
        ]
      },

      "onInput": {
        "events": [
          {
            "eventName": "whatWasInput",
            "condition": "true",
            "actions": [
              {
                "setVariable": {
                  "variableName": "sensorId",
                  "value": "$input.temperatureInput.sensorId"
                }
              },
              {
                "setVariable": {
                  "variableName": "reportedTemperature",
                  "value": "$input.temperatureInput.sensorData.temperature"
                }
              }
            ]
          },
          {
            "eventName": "changeDesired",
            "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
            "actions": [
              {
                "setVariable": {
                  "variableName": "desiredTemperature",
                  "value": "$input.seedTemperatureInput.desiredTemperature"
                }
              }
            ]
          },
          {
            "eventName": "calculateAverage",
```

```
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
            "actions": [
              {
                "setVariable": {
                  "variableName": "averageTemperature",
                  "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
              }
            ]
          },
          {
            "eventName": "areWeThereYet",
            "condition": "(timeout(\"coolingTimer\"))",
            "actions": [
              {
                "setVariable": {
                  "variableName": "goodToGo",
                  "value": "true"
                }
              }
            ]
          }
        ],
        "transitionEvents": [
          {
            "eventName": "anomalousInputArrived",
            "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/anomaly"
                }
              }
            ],
            "nextState": "cooling"
          },

          {
            "eventName": "highTemperatureSpike",
```

```
              "condition": "$input.temperatureInput.sensorData.temperature >
 $variable.rangeHigh",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                  }
                }
              ],
              "nextState": "cooling"
            },

            {
              "eventName": "lowTemperatureSpike",
              "condition": "$input.temperatureInput.sensorData.temperature <
 $variable.rangeLow",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                  }
                },
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
                  }
                },
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/Off"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                  }
                },
                {
                  "setVariable": {
```

```
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "heating"
            },

            {
              "eventName": "desiredTemperature",
              "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <=
($variable.desiredTemperature - $variable.allowedError)) && $variable.goodToGo ==
true",
              "actions": [
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/Off"
                  }
                }
              ],
              "nextState": "idle"
            }
          ]
        }
      },


      {
        "stateName": "heating",
        "onEnter": {
          "events": [
            {
              "eventName": "delay",
              "condition": "!$variable.noDelay && $variable.enteringNewState",
              "actions": [
                {
                  "setTimer": {
                    "timerName": "heatingTimer",
```

```
                "seconds": 120
              }
            },
            {
              "setVariable": {
                "variableName": "goodToGo",
                "value": "false"
              }
            }
          ]
        },
        {
          "eventName": "dontDelay",
          "condition": "$variable.noDelay == true",
          "actions": [
            {
              "setVariable": {
                "variableName": "goodToGo",
                "value": "true"
              }
            }
          ]
        },
        {
          "eventName": "beenHere",
          "condition": "true",
          "actions": [
            {
              "setVariable": {
                "variableName": "enteringNewState",
                "value": "false"
              }
            }
          ]
        }
      ]
    },

    "onInput": {
      "events": [
        {
          "eventName": "whatWasInput",
          "condition": "true",
          "actions": [
```

```
                {
                  "setVariable": {
                    "variableName": "sensorId",
                    "value": "$input.temperatureInput.sensorId"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "reportedTemperature",
                    "value": "$input.temperatureInput.sensorData.temperature"
                  }
                }
              ]
            },
            {
              "eventName": "changeDesired",
              "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "desiredTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                  }
                }
              ]
            },
            {
              "eventName": "calculateAverage",
              "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "averageTemperature",
                    "value": "((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                  }
                }
              ]
            },
            {
              "eventName": "areWeThereYet",
```

```
            "condition": "(timeout(\"heatingTimer\"))",
            "actions": [
              {
                "setVariable": {
                  "variableName": "goodToGo",
                  "value": "true"
                }
              }
            ]
          }
        ],
        "transitionEvents": [
          {
            "eventName": "anomalousInputArrived",
            "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/anomaly"
                }
              }
            ],
            "nextState": "heating"
          },

          {
            "eventName": "highTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
            "actions": [
              {
                "iotTopicPublish": {
                  "mqttTopic": "temperatureSensor/spike"
                }
              },
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                }
              },
              {
                "sns": {
```

```
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/Off"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/On"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                  }
                }
              ],
              "nextState": "cooling"
            },

            {
              "eventName": "lowTemperatureSpike",
              "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                  }
                }
              ],
              "nextState": "heating"
            },

            {
              "eventName": "desiredTemperature",
              "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >=
($variable.desiredTemperature + $variable.allowedError)) && $variable.goodToGo ==
true",
              "actions": [
```

```
                {
                  "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                  }
                },
                {
                  "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/Off"
                  }
                }
              ],
              "nextState": "idle"
            }
          ]
        }
      }

    ],

    "initialStateName": "start"
  },
  "key": "areaId",
  "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}
```

Antwort:

```
{
    "detectorModelConfiguration": {
        "status": "ACTIVATING",
        "lastUpdateTime": 1557523491.168,
        "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
        "creationTime": 1557523491.168,
        "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
areaDetectorModel",
        "key": "areaId",
        "detectorModelName": "areaDetectorModel",
        "detectorModelVersion": "1"
    }
}
```

BatchPutMessageBeispiele

In diesem Beispiel "`BatchPutMessage`" wird verwendet, um eine Melderinstanz für einen Bereich zu erstellen und die anfänglichen Betriebsparameter zu definieren.

Verwendeter CLI-Befehl:

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json --cli-
binary-format raw-in-base64-out
```

Datei: `seedExample.json`

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 20.0, \"allowedError
\": 0.7, \"rangeHigh\": 30.0, \"rangeLow\": 15.0, \"anomalousHigh\": 60.0,
 \"anomalousLow\": 0.0, \"sensorCount\": 10, \"noDelay\": false}"
    }
  ]
}
```

Antwort:

```
{
    "BatchPutMessageErrorEntries": []
}
```

In diesem Beispiel "`BatchPutMessage`" wird es verwendet, um Temperatursensorwerte für einen einzelnen Sensor in einem Bereich zu melden.

Verwendeter CLI-Befehl:

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json --
cli-binary-format raw-in-base64-out
```

Datei: `temperatureExample.json`

```
{
```

```
  "messages": [
    {
      "messageId": "00005",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"05\", \"areaId\": \"Area51\", \"sensorData\":
 {\"temperature\": 23.12} }"
    }
  ]
}
```

Antwort:

```
{
    "BatchPutMessageErrorEntries": []
}
```

In diesem Beispiel "`BatchPutMessage`" wird verwendet, um die gewünschte Temperatur für einen Bereich zu ändern.

Verwendeter CLI-Befehl:

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json --
cli-binary-format raw-in-base64-out
```

Datei: `seedSetDesiredTemp.json`

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

Antwort:

```
{
    "BatchPutMessageErrorEntries": []
}
```

Beispiele für Amazon SNS SNS-Nachrichten, die von der `Area51` Detector-Instance generiert wurden:

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{
      "detectorModelName":"areaDetectorModel",
      "keyValue":"Area51",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"seedTemperatureInput",
      "messageId":"00001",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"start",
      "variables":{
        "sensorCount":10,
        "rangeHigh":30.0,
        "resetMe":false,
        "enteringNewState":true,
        "averageTemperature":20.0,
        "rangeLow":15.0,
        "noDelay":false,
        "allowedError":0.7,
        "desiredTemperature":20.0,
        "anomalousHigh":60.0,
        "reportedTemperature":0.1,
        "anomalousLow":0.0,
        "sensorId":0
      },
      "timers":{}
    }
  },
  "eventName":"resetHeatCool"
}
```

```
Cooling system off command> {
```

```
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192",
    "detector":{
      "detectorModelName":"areaDetectorModel",
      "keyValue":"Area51",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"seedTemperatureInput",
      "messageId":"00001",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"start",
      "variables":{
        "sensorCount":10,
        "rangeHigh":30.0,
        "resetMe":false,
        "enteringNewState":true,
        "averageTemperature":20.0,
        "rangeLow":15.0,
        "noDelay":false,
        "allowedError":0.7,
        "desiredTemperature":20.0,
        "anomalousHigh":60.0,
        "reportedTemperature":0.1,
        "anomalousLow":0.0,
        "sensorId":0
      },
      "timers":{}
    }
  },
  "eventName":"resetHeatCool"
}
```


In diesem Beispiel verwenden wir die "`DescribeDetector`" API, um Informationen über den aktuellen Status einer Detector-Instance abzurufen.

```
aws iotevents-data describe-detector --detector-model-name areaDetectorModel --key-
value Area51
```

Antwort:

```
{
    "detector": {
        "lastUpdateTime": 1557521572.216,
        "creationTime": 1557520274.405,
        "state": {
            "variables": [
                {
                    "name": "resetMe",
                    "value": "false"
                },
                {
                    "name": "rangeLow",
                    "value": "15.0"
                },
                {
                    "name": "noDelay",
                    "value": "false"
                },
                {
                    "name": "desiredTemperature",
                    "value": "20.0"
                },
                {
                    "name": "anomalousLow",
                    "value": "0.0"
                },
                {
                    "name": "sensorId",
                    "value": "\"01\""
                },
                {
                    "name": "sensorCount",
                    "value": "10"
                },
                {
                    "name": "rangeHigh",
                    "value": "30.0"
                },
                {
                    "name": "enteringNewState",
                    "value": "false"
                },
                {
```

```
                    "name": "averageTemperature",
                    "value": "19.572"
                },
                {
                    "name": "allowedError",
                    "value": "0.7"
                },
                {
                    "name": "anomalousHigh",
                    "value": "60.0"
                },
                {
                    "name": "reportedTemperature",
                    "value": "15.72"
                },
                {
                    "name": "goodToGo",
                    "value": "false"
                }
            ],
            "stateName": "idle",
            "timers": [
                {
                    "timestamp": 1557520454.0,
                    "name": "idleTimer"
                }
            ]
        },
        "keyValue": "Area51",
        "detectorModelName": "areaDetectorModel",
        "detectorModelVersion": "1"
    }
}
```

BatchUpdateDetector Beispiel

In diesem Beispiel "`BatchUpdateDetector`" wird es verwendet, um Betriebsparameter für eine funktionierende Melderinstanz zu ändern.

Verwendeter CLI-Befehl:

```
aws iotevents-data batch-update-detector --cli-input-json file://areaDM.BUD.json
```

Datei: `areaDM.BUD.json`

```
{
  "detectors": [
    {
      "messageId": "0001",
      "detectorModelName": "areaDetectorModel",
      "keyValue": "Area51",
      "state": {
        "stateName": "start",
        "variables": [
          {
            "name": "desiredTemperature",
            "value": "22"
          },
          {
            "name": "averageTemperature",
            "value": "22"
          },
          {
            "name": "allowedError",
            "value": "1.0"
          },
          {
            "name": "rangeHigh",
            "value": "30.0"
          },
          {
            "name": "rangeLow",
            "value": "15.0"
          },
          {
            "name": "anomalousHigh",
            "value": "60.0"
          },
          {
            "name": "anomalousLow",
            "value": "0.0"
          },
          {
            "name": "sensorCount",
            "value": "12"
          },
          {
```

```
            "name": "noDelay",
            "value": "true"
          },
          {
            "name": "goodToGo",
            "value": "true"
          },
          {
            "name": "sensorId",
            "value": "0"
          },
          {
            "name": "reportedTemperature",
            "value": "0.1"
          },
          {
            "name": "resetMe",
            "value": "true"
          }
        ],
        "timers": [
        ]
      }
    }
  ]
}
```

Antwort:

```
{
    An error occurred (InvalidRequestException) when calling the BatchUpdateDetector
 operation: Number of variables in the detector exceeds the limit 10
}
```

AWS IoT CoreBeispiele für die Regel-Engine

Die folgenden Regeln veröffentlichen AWS IoT Events MQTT-Nachrichten erneut als Shadow-Update-Anforderungsnachrichten. Wir gehen davon aus, dass für jeden Bereich, der durch das Detektormodell gesteuert wird, AWS IoT Core Dinge für eine Heiz- und eine Kühleinheit definiert sind.

In diesem Beispiel haben wir Dinge mit dem Namen `"Area51HeatingUnit"` und definiert`"Area51CoolingUnit"`.

Verwendeter CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOffRule.json
```

Datei: ADMShadowCoolOffRule.json

```
{
  "ruleName": "ADMShadowCoolOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/Off'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

Verwendeter CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOnRule.json
```

Datei: ADMShadowCoolOnRule.json

```
{
  "ruleName": "ADMShadowCoolOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/On'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
```

```
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

Verwendeter CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowHeatOffRule.json
```

Datei: `ADMShadowHeatOffRule.json`

```
{
  "ruleName": "ADMShadowHeatOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/Off'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

Verwendeter CLI-Befehl:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowHeatOnRule.json
```

Datei: `ADMShadowHeatOnRule.json`

```
{
  "ruleName": "ADMShadowHeatOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/On'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
 request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Antwort: [leer]

Kräne

Hintergrundgeschichte

Ein Betreiber vieler Krane möchte erkennen, wann die Maschinen gewartet oder ausgetauscht werden müssen, und entsprechende Benachrichtigungen auslösen. Jeder Kran hat einen Motor. Ein Motor sendet Nachrichten (Eingänge) mit Informationen über Druck und Temperatur aus. Der Betreiber benötigt zwei Stufen von Ereignismeldern:

- Ein Ereignismelder auf Kranebene
- Ein Ereignismelder auf Motorebene

Mithilfe von Meldungen von den Motoren (die Metadaten sowohl mit dem `"craneId"` als auch mit enthalten`"motorId"`) kann der Bediener die Ereignismelder auf beiden Ebenen mithilfe

eines geeigneten Routings ausführen. Wenn die Veranstaltungsbedingungen erfüllt sind, sollten Benachrichtigungen an die entsprechenden Amazon SNS SNS-Themen gesendet werden. Der Betreiber kann die Meldermodelle so konfigurieren, dass keine doppelten Benachrichtigungen ausgelöst werden.

Dieses Beispiel demonstriert die folgenden Funktionsfähigkeiten:

- Eingaben erstellen, lesen, aktualisieren, löschen (CRUD).
- Erstellen, Lesen, Aktualisieren, Löschen (CRUD) von Ereignismeldermodellen und verschiedenen Versionen von Ereignismeldern.
- Weiterleitung eines Eingangs an mehrere Ereignismelder.
- Aufnahme von Eingaben in ein Detektormodell.
- Bewertung von Triggerbedingungen und Lebenszyklusereignissen.
- Fähigkeit, in Bedingungen auf Zustandsvariablen zu verweisen und ihre Werte in Abhängigkeit von den Bedingungen festzulegen.
- Laufzeitorchestrierung mit Definition, Status, Trigger-Evaluator und Aktionsausführung.
- Ausführung von Aktionen `ActionsExecutor` mit einem SNS-Ziel.

Befehle

```
#Create Pressure Input
aws iotevents create-input  --cli-input-json file://pressureInput.json
aws iotevents describe-input --input-name PressureInput
aws iotevents update-input  --cli-input-json file://pressureInput.json
aws iotevents list-inputs
aws iotevents delete-input --input-name PressureInput

#Create Temperature Input
aws iotevents create-input  --cli-input-json file://temperatureInput.json
aws iotevents describe-input --input-name TemperatureInput
aws iotevents update-input  --cli-input-json file://temperatureInput.json
aws iotevents list-inputs
aws iotevents delete-input --input-name TemperatureInput

#Create Motor Event Detector using pressure and temperature input
aws iotevents create-detector-model  --cli-input-json file://motorDetectorModel.json
aws iotevents describe-detector-model --detector-model-name motorDetectorModel
```

```
aws iotevents update-detector-model  --cli-input-json file://
updateMotorDetectorModel.json
aws iotevents list-detector-models
aws iotevents list-detector-model-versions --detector-model-name motorDetectorModel
aws iotevents delete-detector-model --detector-model-name motorDetectorModel

#Create Crane Event Detector using temperature input
aws iotevents create-detector-model  --cli-input-json file://craneDetectorModel.json
aws iotevents describe-detector-model --detector-model-name craneDetectorModel
aws iotevents update-detector-model  --cli-input-json file://
updateCraneDetectorModel.json
aws iotevents list-detector-models
aws iotevents list-detector-model-versions --detector-model-name craneDetectorModel
aws iotevents delete-detector-model --detector-model-name craneDetectorModel

#Replace craneIds
sed -i '' "s/100008/100009/g" messages/*

#Replace motorIds
sed -i '' "s/200008/200009/g" messages/*

#Send HighPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highPressureMessage.json --cli-binary-format raw-in-base64-out

#Send HighTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highTemperatureMessage.json --cli-binary-format raw-in-base64-out

#Send LowPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowPressureMessage.json --cli-binary-format raw-in-base64-out

#Send LowTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowTemperatureMessage.json --cli-binary-format raw-in-base64-out
```

Detektormodelle

Datei: `craneDetectorModel.json`

```
{
```

```
    "detectorModelName": "craneDetectorModel",
    "detectorModelDefinition": {
        "states": [
            {
                "stateName": "Running",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "init",
                            "condition": "true",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "0"
                                    }
                                }
                            ]
                        }
                    ]
                },
                "onInput": {
                    "events": [
                        {
                            "eventName": "Overheated",
                            "condition": "$input.TemperatureInput.temperature > 35",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "$variable.craneThresholdBreached + 1"
                                    }
                                }
                            ]
                        },
                        {
                            "eventName": "Crane Threshold Breached",
                            "condition": "$variable.craneThresholdBreached > 5",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
                                    }
```

```
                            }
                        ]
                    },
                    {
                        "eventName": "Underheated",
                        "condition": "$input.TemperatureInput.temperature < 25",
                        "actions": [
                            {
                                "setVariable": {
                                    "variableName": "craneThresholdBreached",
                                    "value": "0"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ],
    "initialStateName": "Running"
},
"key": "craneid",
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}
```

Um ein vorhandenes Meldermodell zu aktualisieren. Datei: `updateCraneDetectorModel.json`

```
{
    "detectorModelName": "craneDetectorModel",
    "detectorModelDefinition": {
        "states": [
            {
                "stateName": "Running",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "init",
                            "condition": "true",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "0"
```

```
                                    }
                                },
                                {
                                    "setVariable": {
                                        "variableName": "alarmRaised",
                                        "value": "'false'"
                                    }
                                }
                            ]
                        }
                    ]
                },
                "onInput": {
                    "events": [
                        {
                            "eventName": "Overheated",
                            "condition": "$input.TemperatureInput.temperature > 30",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "$variable.craneThresholdBreached + 1"
                                    }
                                }
                            ]
                        },
                        {
                            "eventName": "Crane Threshold Breached",
                            "condition": "$variable.craneThresholdBreached > 5 &&
 $variable.alarmRaised == 'false'",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
                                    }
                                },
                                {
                                    "setVariable": {
                                        "variableName": "alarmRaised",
                                        "value": "'true'"
                                    }
                                }
                            ]
```

```
                        },
                        {
                            "eventName": "Underheated",
                            "condition": "$input.TemperatureInput.temperature < 10",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "craneThresholdBreached",
                                        "value": "0"
                                    }
                                }
                            ]
                        }
                    ]
                }
            }
        ],
        "initialStateName": "Running"
    },
    "roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}
```

Datei: `motorDetectorModel.json`

```
{
    "detectorModelName": "motorDetectorModel",
    "detectorModelDefinition": {
        "states": [
            {
                "stateName": "Running",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "init",
                            "condition": "true",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "motorThresholdBreached",
                                        "value": "0"
                                    }
                                }
                            ]
```

```
                        }
                    ]
                },
                "onInput": {
                    "events": [
                        {
                            "eventName": "Overheated And Overpressurized",
                            "condition": "$input.PressureInput.pressure > 70 &&
 $input.TemperatureInput.temperature > 30",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "motorThresholdBreached",
                                        "value": "$variable.motorThresholdBreached + 1"
                                    }
                                }
                            ]
                        },
                        {
                            "eventName": "Motor Threshold Breached",
                            "condition": "$variable.motorThresholdBreached > 5",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
                                    }
                                }
                            ]
                        }
                    ]
                }
            }
        ],
        "initialStateName": "Running"
    },
    "key": "motorid",
    "roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}
```

Um ein vorhandenes Meldermodell zu aktualisieren. Datei: `updateMotorDetectorModel.json`

```
{
```

```
    "detectorModelName": "motorDetectorModel",
    "detectorModelDefinition": {
        "states": [
            {
                "stateName": "Running",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "init",
                            "condition": "true",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "motorThresholdBreached",
                                        "value": "0"
                                    }
                                }
                            ]
                        }
                    ]
                },
                "onInput": {
                    "events": [
                        {
                            "eventName": "Overheated And Overpressurized",
                            "condition": "$input.PressureInput.pressure > 70 &&
 $input.TemperatureInput.temperature > 30",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "motorThresholdBreached",
                                        "value": "$variable.motorThresholdBreached + 1"
                                    }
                                }
                            ]
                        },
                        {
                            "eventName": "Motor Threshold Breached",
                            "condition": "$variable.motorThresholdBreached > 5",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
```

```
                                    }
                                }
                            ]
                        }
                    ]
                }
            }
        ],
        "initialStateName": "Running"
    },
    "roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}
```

Eingaben

Datei: `pressureInput.json`

```
{
    "inputName": "PressureInput",
    "inputDescription": "this is a pressure input description",
    "inputDefinition": {
        "attributes": [
          {"jsonPath": "pressure"}
        ]
    }
}
```

Datei: `temperatureInput.json`

```
{
    "inputName": "TemperatureInput",
    "inputDescription": "this is temperature input description",
    "inputDefinition": {
        "attributes": [
            {"jsonPath": "temperature"}
        ]
    }
}
```

Nachrichten

Datei: `highPressureMessage.json`

```
{
   "messages": [
        {
           "messageId": "1",
           "inputName": "PressureInput",
           "payload": "{\"craneid\": \"100009\", \"pressure\": 80, \"motorid\":
 \"200009\"}"

        }
    ]
}
```

Datei: `highTemperatureMessage.json`

```
{
   "messages": [
        {
           "messageId": "2",
           "inputName": "TemperatureInput",
           "payload": "{\"craneid\": \"100009\", \"temperature\": 40, \"motorid\":
 \"200009\"}"
        }
    ]
}
```

Datei: `lowPressureMessage.json`

```
{
   "messages": [
        {
           "messageId": "1",
           "inputName": "PressureInput",
           "payload": "{\"craneid\": \"100009\", \"pressure\": 20, \"motorid\":
 \"200009\"}"
        }
    ]
}
```

Datei: `lowTemperatureMessage.json`

```
{
   "messages": [
```

```
        {
           "messageId": "2",
           "inputName": "TemperatureInput",
           "payload": "{\"craneid\": \"100009\", \"temperature\": 20, \"motorid\":
 \"200009\"}"
        }
    ]
}
```

Erkennung von Ereignissen mit Sensoren und Anwendungen

Dieses Detektormodell ist eine der Vorlagen, die in der AWS IoT Events Konsole verfügbar sind. Es ist hier der Einfachheit halber enthalten.

```
{
    "detectorModelName": "EventDetectionSensorsAndApplications",
    "detectorModelDefinition": {
        "states": [
            {
                "onInput": {
                    "transitionEvents": [],
                    "events": []
                },
                "stateName": "Device_exception",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Send_mqtt",
                            "actions": [
                                {
                                    "iotTopicPublish": {
                                        "mqttTopic": "Device_stolen"
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
```

```
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "To_in_use",
                            "actions": [],
                            "condition": "$variable.position !=
$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.gps_position",
                            "nextState": "Device_in_use"
                        }
                    ],
                    "events": []
                },
                "stateName": "Device_idle",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Set_position",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "position",
                                        "value":
"$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.gps_position"
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "To_exception",
                            "actions": [],
```

```
                            "condition":
 "$input.AWS_IoTEvents_Blueprints_Tracking_UserInput.device_id !=
 $input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.device_id",
                            "nextState": "Device_exception"
                        }
                    ],
                    "events": []
                },
                "stateName": "Device_in_use",
                "onEnter": {
                    "events": []
                },
                "onExit": {
                    "events": []
                }
            }
        ],
        "initialStateName": "Device_idle"
    }
}
```

Gerät HeartBeat

Dieses Meldermodell ist eine der Vorlagen, die in der AWS IoT Events Konsole verfügbar sind. Es ist hier der Einfachheit halber enthalten.

```
{
    "detectorModelDefinition": {
        "states": [
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "To_normal",
                            "actions": [],
                            "condition":
 "currentInput(\"AWS_IoTEvents_Blueprints_Heartbeat_Input\")",
                            "nextState": "Normal"
                        }
                    ],
                    "events": []
                },
```

```
                "stateName": "Offline",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Send_notification",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "sns-topic-arn"
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "Go_offline",
                            "actions": [],
                            "condition": "timeout(\"awake\")",
                            "nextState": "Offline"
                        }
                    ],
                    "events": [
                        {
                            "eventName": "Reset_timer",
                            "actions": [
                                {
                                    "resetTimer": {
                                        "timerName": "awake"
                                    }
                                }
                            ],
                            "condition":
"currentInput(\"AWS_IoTEvents_Blueprints_Heartbeat_Input\")"
                        }
                    ]
```

```
                },
                "stateName": "Normal",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Create_timer",
                            "actions": [
                                {
                                    "setTimer": {
                                        "seconds": 300,
                                        "timerName": "awake"
                                    }
                                }
                            ],
                            "condition":
 "$input.AWS_IoTEvents_Blueprints_Heartbeat_Input.value > 0"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            }
        ],
        "initialStateName": "Normal"
    }
}
```

ISA Alarm

Dieses Meldermodell ist eine der Vorlagen, die in der AWS IoT Events Konsole verfügbar sind. Es ist hier der Einfachheit halber enthalten.

```
{
    "detectorModelName": "AWS_IoTEvents_Blueprints_ISA_Alarm",
    "detectorModelDefinition": {
        "states": [
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "unshelve",
```

```
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"rtnunack\"",
                            "nextState": "RTN_Unacknowledged"
                        },
                        {
                            "eventName": "unshelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"ack\"",
                            "nextState": "Acknowledged"
                        },
                        {
                            "eventName": "unshelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"unack\"",
                            "nextState": "Unacknowledged"
                        },
                        {
                            "eventName": "unshelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"normal\"",
                            "nextState": "Normal"
                        }
                    ],
                    "events": []
                },
                "stateName": "Shelved",
                "onEnter": {
                    "events": []
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
```

```
                        {
                            "eventName": "abnormal_condition",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value > $variable.higher_threshold ||
$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value < $variable.lower_threshold",
                            "nextState": "Unacknowledged"
                        },
                        {
                            "eventName": "acknowledge",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"acknowledge\"",
                            "nextState": "Normal"
                        },
                        {
                            "eventName": "shelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
                            "nextState": "Shelved"
                        },
                        {
                            "eventName": "remove_from_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
                            "nextState": "Out_of_service"
                        },
                        {
                            "eventName": "suppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
                            "nextState": "Suppressed_by_design"
                        }
                    ],
                    "events": []
                },
                "stateName": "RTN_Unacknowledged",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "State Save",
```

```
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "state",
                                        "value": "\"rtnunack\""
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "abnormal_condition",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value > $variable.higher_threshold ||
$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value < $variable.lower_threshold",
                            "nextState": "Unacknowledged"
                        },
                        {
                            "eventName": "shelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
                            "nextState": "Shelved"
                        },
                        {
                            "eventName": "remove_from_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
                            "nextState": "Out_of_service"
                        },
                        {
                            "eventName": "suppression",
                            "actions": [],
```

```
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
                            "nextState": "Suppressed_by_design"
                        }
                    ],
                    "events": [
                        {
                            "eventName": "Create Config variables",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "lower_threshold",
                                        "value":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.lower_threshold"
                                    }
                                },
                                {
                                    "setVariable": {
                                        "variableName": "higher_threshold",
                                        "value":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.higher_threshold"
                                    }
                                }
                            ],
                            "condition": "$variable.lower_threshold !=
$variable.lower_threshold"
                        }
                    ]
                },
                "stateName": "Normal",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "State Save",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "state",
                                        "value": "\"normal\""
                                    }
                                }
                            ],
                            "condition": "true"
                        }
```

```
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "acknowledge",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"acknowledge\"",
                            "nextState": "Acknowledged"
                        },
                        {
                            "eventName": "return_to_normal",
                            "actions": [],
                            "condition":
"($input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value <= $variable.higher_threshold
&& $input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value >=
$variable.lower_threshold)",
                            "nextState": "RTN_Unacknowledged"
                        },
                        {
                            "eventName": "shelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
                            "nextState": "Shelved"
                        },
                        {
                            "eventName": "remove_from_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
                            "nextState": "Out_of_service"
                        },
                        {
                            "eventName": "suppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
```

```
                            "nextState": "Suppressed_by_design"
                        }
                    ],
                    "events": []
                },
                "stateName": "Unacknowledged",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "State Save",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "state",
                                        "value": "\"unack\""
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "unsuppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"normal\"",
                            "nextState": "Normal"
                        },
                        {
                            "eventName": "unsuppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"unack\"",
                            "nextState": "Unacknowledged"
```

```
                        },
                        {
                            "eventName": "unsuppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"ack\"",
                            "nextState": "Acknowledged"
                        },
                        {
                            "eventName": "unsuppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"rtnunack\"",
                            "nextState": "RTN_Unacknowledged"
                        }
                    ],
                    "events": []
                },
                "stateName": "Suppressed_by_design",
                "onEnter": {
                    "events": []
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "return_to_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"rtnunack\"",
                            "nextState": "RTN_Unacknowledged"
                        },
                        {
                            "eventName": "return_to_service",
                            "actions": [],
```

```
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"unack\"",
                            "nextState": "Unacknowledged"
                        },
                        {
                            "eventName": "return_to_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"ack\"",
                            "nextState": "Acknowledged"
                        },
                        {
                            "eventName": "return_to_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"normal\"",
                            "nextState": "Normal"
                        }
                    ],
                    "events": []
                },
                "stateName": "Out_of_service",
                "onEnter": {
                    "events": []
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "re-alarm",
                            "actions": [],
                            "condition": "timeout(\"snooze\")",
                            "nextState": "Unacknowledged"
                        },
                        {
                            "eventName": "return_to_normal",
                            "actions": [],
```

```
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"reset\"",
                            "nextState": "Normal"
                        },
                        {
                            "eventName": "shelve",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
                            "nextState": "Shelved"
                        },
                        {
                            "eventName": "remove_from_service",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
                            "nextState": "Out_of_service"
                        },
                        {
                            "eventName": "suppression",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
                            "nextState": "Suppressed_by_design"
                        }
                    ],
                    "events": []
                },
                "stateName": "Acknowledged",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Create Timer",
                            "actions": [
                                {
                                    "setTimer": {
                                        "seconds": 60,
                                        "timerName": "snooze"
                                    }
                                }
                            ],
                            "condition": "true"
                        },
                        {
```

```
                            "eventName": "State Save",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "state",
                                        "value": "\"ack\""
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            }
        ],
        "initialStateName": "Normal"
    },
    "detectorModelDescription": "This detector model is used to detect if a monitored
 device is in an Alarming State in accordance to the ISA 18.2.",
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "key": "alarmId"
}
```

Einfacher Alarm

Dieses Meldermodell ist eine der Vorlagen, die in der AWS IoT Events Konsole verfügbar sind. Es ist hier der Einfachheit halber enthalten.

```
{
    "detectorModelDefinition": {
        "states": [
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "not_fixed",
                            "actions": [],
                            "condition": "timeout(\"snoozeTime\")",
                            "nextState": "Alarming"
```

```
                        },
                        {
                            "eventName": "reset",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"reset\"",
                            "nextState": "Normal"
                        }
                    ],
                    "events": [
                        {
                            "eventName": "DND",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "dnd_active",
                                        "value": "1"
                                    }
                                }
                            ],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"dnd\""
                        }
                    ]
                },
                "stateName": "Snooze",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Create Timer",
                            "actions": [
                                {
                                    "setTimer": {
                                        "seconds": 120,
                                        "timerName": "snoozeTime"
                                    }
                                }
                            ],
                            "condition": "true"
                        }
                    ]
                },
                "onExit": {
                    "events": []
```

```
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "out_of_range",
                            "actions": [],
                            "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.value > $variable.threshold",
                            "nextState": "Alarming"
                        }
                    ],
                    "events": [
                        {
                            "eventName": "Create Config variables",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "threshold",
                                        "value":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.threshold"
                                    }
                                }
                            ],
                            "condition": "$variable.threshold != $variable.threshold"
                        }
                    ]
                },
                "stateName": "Normal",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Init",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "dnd_active",
                                        "value": "0"
                                    }
                                }
                            ],
                            "condition": "true"
                        }
```

```
                    ]
                },
                "onExit": {
                    "events": []
                }
            },
            {
                "onInput": {
                    "transitionEvents": [
                        {
                            "eventName": "reset",
                            "actions": [],
                            "condition":
 "$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"reset\"",
                            "nextState": "Normal"
                        },
                        {
                            "eventName": "acknowledge",
                            "actions": [],
                            "condition":
 "$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"acknowledge\"",
                            "nextState": "Snooze"
                        }
                    ],
                    "events": [
                        {
                            "eventName": "Escalated Alarm Notification",
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:escalatedAlarmNotification"
                                    }
                                }
                            ],
                            "condition": "timeout(\"unacknowledgeTIme\")"
                        }
                    ]
                },
                "stateName": "Alarming",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Alarm Notification",
```

```
                            "actions": [
                                {
                                    "sns": {
                                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:alarmNotification"
                                    }
                                },
                                {
                                    "setTimer": {
                                        "seconds": 300,
                                        "timerName": "unacknowledgeTIme"
                                    }
                                }
                            ],
                            "condition": "$variable.dnd_active != 1"
                        }
                    ]
                },
                "onExit": {
                    "events": []
                }
            }
        ],
        "initialStateName": "Normal"
    },
    "detectorModelDescription": "This detector model is used to detect if a monitored
 device is in an Alarming State.",
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "key": "alarmId"
}
```

Überwachung mit -Alarmen

AWS IoT EventsMithilfe von Alarmen können Sie Ihre Daten auf Änderungen überwachen. Bei den Daten kann es sich um Kennzahlen handeln, die Sie für Ihre Ausrüstung und Prozesse messen. Sie können Alarme erstellen, die Benachrichtigungen senden, wenn ein Schwellenwert überschritten wird. Alarme helfen Ihnen dabei, Probleme zu erkennen, die Wartung zu rationalisieren und die Leistung Ihrer Geräte und Prozesse zu optimieren.

Alarme sind Beispiele von Alarmmodellen. Das Alarmmodell legt fest, was erkannt werden soll, wann Benachrichtigungen gesendet werden sollen, wer benachrichtigt wird und vieles mehr. Sie können auch eine oder mehrere unterstützte Aktionen angeben, die ausgeführt werden, wenn sich der Alarmstatus ändert. AWS IoT Eventsleitet aus Ihren Daten abgeleitete Eingabeattribute an die entsprechenden Alarme weiter. Wenn die Daten, die Sie überwachen, außerhalb des angegebenen Bereichs liegen, wird der Alarm ausgelöst. Sie können die Alarme auch bestätigen oder sie in den Schlummermodus versetzen.

Arbeiten mit AWS IoT SiteWise

Sie können AWS IoT Events Alarme verwenden, um die Eigenschaften von Vermögenswerten in zu überwachenAWS IoT SiteWise. AWS IoT SiteWisesendet Eigenschaftswerte von Vermögenswerten an AWS IoT Events Alarme. AWS IoT Eventssendet den Alarmstatus anAWS IoT SiteWise.

AWS IoT SiteWiseunterstützt auch externe Alarme. Sie können externe Alarme wählen, wenn Sie Alarme außerhalb von verwenden AWS IoT SiteWise und über eine Lösung verfügen, die Alarmstatusdaten zurückgibt. Der externe Alarm enthält eine Messeigenschaft, die die Alarmzustandsdaten aufnimmt.

AWS IoT SiteWisewertet den Status externer Alarme nicht aus. Außerdem können Sie einen externen Alarm nicht bestätigen oder deaktivieren, wenn sich der Alarmstatus ändert.

Sie können die SiteWise Überwachungsfunktion verwenden, um den Status externer Alarme in SiteWise Monitor-Portalen einzusehen.

Weitere Informationen finden Sie unter Überwachen von Daten mit Alarmen im AWS IoT SiteWiseBenutzerhandbuch und Überwachen mit Alarmen im SiteWise Monitor-Anwendungshandbuch.

Bestätigen Sie den Ablauf

Wenn Sie ein Alarmmodell erstellen, wählen Sie aus, ob der Bestätigungsfluss aktiviert werden soll. Wenn Sie den Bestätigungsfluss aktivieren, wird Ihr Team benachrichtigt, wenn sich der Alarmstatus ändert. Ihr Team kann den Alarm bestätigen und eine Notiz hinterlassen. Sie können beispielsweise die Informationen zum Alarm und die Maßnahmen, die Sie ergreifen werden, um das Problem zu beheben, angeben. Wenn die Daten, die Sie überwachen, außerhalb des angegebenen Bereichs liegen, wird der Alarm ausgelöst.

Alarme haben die folgenden Status:

`DISABLED`

Wenn sich der Alarm im `DISABLED` Status befindet, ist er nicht bereit, Daten auszuwerten. Um den Alarm zu aktivieren, müssen Sie den Alarm in den `NORMAL` Status ändern.

`NORMAL`

Wenn sich der Alarm im `NORMAL` Status befindet, ist er bereit, Daten auszuwerten.

`ACTIVE`

Befindet sich der Alarm im `ACTIVE` Status, wird der Alarm ausgelöst. Die Daten, die Sie überwachen, liegen außerhalb des angegebenen Bereichs.

`ACKNOWLEDGED`

Wenn sich der Alarm im `ACKNOWLEDGED` Status befindet, wurde der Alarm ausgelöst und Sie haben den Alarm bestätigt.

`LATCHED`

Der Alarm wurde ausgelöst, aber Sie haben den Alarm nach einer gewissen Zeit nicht bestätigt. Der Alarm wechselt automatisch in den `NORMAL` Status.

`SNOOZE_DISABLED`

Wenn sich der Alarm im `SNOOZE_DISABLED` Status befindet, ist der Alarm für einen bestimmten Zeitraum deaktiviert. Nach Ablauf der Schlummerzeit wechselt der Alarm automatisch in den `NORMAL` Status.

Ein Alarmmodell erstellen

Sie können AWS IoT Events Alarme verwenden, um Ihre Daten zu überwachen und sich benachrichtigen zu lassen, wenn ein Schwellenwert überschritten wird. Alarme stellen Parameter bereit, die Sie verwenden, um ein Alarmmodell zu erstellen oder zu konfigurieren. Sie können die AWS IoT Events Konsole oder AWS IoT Events API verwenden, um das Alarmmodell zu erstellen oder zu konfigurieren. Wenn Sie das Alarmmodell konfigurieren, werden Änderungen wirksam, sobald neue Daten eintreffen.

Voraussetzungen

Die folgenden Anforderungen gelten, wenn Sie ein Alarmmodell erstellen.

- Sie können ein Alarmmodell erstellen, um ein Eingabeattribut AWS IoT Events oder eine Anlageneigenschaft in zu überwachenAWS IoT SiteWise.
  - Wenn Sie ein Eingabeattribut in überwachen möchtenAWS IoT Events, gehen Sie wie folgt vor, bevor Sie das Alarmmodell erstellen:
    - Schritt 1: Lesen Sie die Übersicht unter Eingabe erstellen.
    - Schritt 2: Lesen Sie die Anweisungen zum Erstellen einer Eingabe im Navigationsbereich.
  - Wenn Sie sich für die Überwachung einer Anlageneigenschaft entscheiden, müssen Sie zunächst ein Anlagenmodell erstellen, AWS IoT SiteWise bevor Sie das Alarmmodell erstellen.
- Sie müssen über eine IAM-Rolle verfügen, die es Ihrem Alarm ermöglicht, Aktionen auszuführen und auf AWS Ressourcen zuzugreifen. Weitere Informationen finden Sie unter Berechtigungen einrichten für AWS IoT Events.
- Alle AWS Ressourcen, die in diesem Tutorial verwendet werden, müssen sich in derselben AWS Region befinden.

Ein Alarmmodell (Konsole) erstellen

Im Folgenden wird gezeigt, wie Sie ein Alarmmodell zur Überwachung eines AWS IoT Events Attributs in der AWS IoT Events Konsole erstellen.

1. Melden Sie sich an der AWS IoT Events-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Alarmmodelle aus.
3. Wählen Sie auf der Seite Alarmmodelle die Option Alarmmodell erstellen aus.

4. Gehen Sie im Abschnitt Details zum Alarmmodell wie folgt vor:

   a. Geben Sie einen eindeutigen Namen ein.

   b. (Optional) Geben Sie eine Beschreibung ein.

5. Gehen Sie im Bereich Alarmziel wie folgt vor:

   > **Important**
   >
   > Wenn Sie sich für eine AWS IoT SiteWiseAnlageneigenschaft entscheiden, müssen Sie in ein Asset-Modell erstellt habenAWS IoT SiteWise.

   a. Wählen Sie das AWS IoT Events Eingabeattribut.

   b. Wählen Sie die Eingabe aus.

   c. Wählen Sie den Eingabeattributschlüssel aus. Dieses Eingabeattribut wird als Schlüssel zum Erstellen des Alarms verwendet. AWS IoT Eventsleitet die mit diesem Schlüssel verknüpften Eingaben an den Alarm weiter.

      > **Important**
      >
      > Wenn die Nutzlast der Eingabenachricht diesen Eingabeattributschlüssel nicht enthält oder wenn sich der Schlüssel nicht in demselben JSON-Pfad befindet, der im Schlüssel angegeben ist, schlägt die Eingabe in der Nachricht fehl. AWS IoT Events

6. Im Abschnitt Schwellenwertdefinitionen definieren Sie das Eingabeattribut, den Schwellenwert und den Vergleichsoperator, der AWS IoT Events verwendet wird, um den Status des Alarms zu ändern.

   a. Wählen Sie unter Eingabeattribut das Attribut aus, das Sie überwachen möchten.

      Jedes Mal, wenn dieses Eingabeattribut neue Daten empfängt, werden diese ausgewertet, um den Status des Alarms zu bestimmen.

   b. Wählen Sie unter Operator den Vergleichsoperator aus. Der Operator vergleicht Ihr Eingabeattribut mit dem Schwellenwert für Ihr Attribut.

      Sie können aus diesen Optionen wählen:

      - > größer als

- >= größer als oder gleich
- < kleiner als
- <= kleiner oder gleich
- = gleich
- ! = nicht gleich

c. Geben Sie für den Schwellenwert eine Zahl ein, oder wählen Sie ein Attribut in den AWS IoT Events Eingaben aus. AWS IoT Eventsvergleicht diesen Wert mit dem Wert des von Ihnen ausgewählten Eingabeattributs.

d. (Optional) Verwenden Sie für Schweregrad eine Zahl, von der Ihr Team weiß, dass sie den Schweregrad dieses Alarms wiedergibt.

7. (Optional) Konfigurieren Sie im Abschnitt Benachrichtigungseinstellungen die Benachrichtigungseinstellungen für den Alarm.

   Sie können bis zu 10 Benachrichtigungen hinzufügen. Gehen Sie für Benachrichtigung 1 wie folgt vor:

   a. Wählen Sie für Protokoll eine der folgenden Optionen aus:

      - E-Mail und Text — Der Alarm sendet eine SMS-Benachrichtigung und eine E-Mail-Benachrichtigung.
      - E-Mail — Der Alarm sendet eine E-Mail-Benachrichtigung.
      - Text — Der Alarm sendet eine SMS-Benachrichtigung.

   b. Geben Sie als Absender die E-Mail-Adresse an, an die Benachrichtigungen zu diesem Alarm gesendet werden können.

      Um Ihrer Absenderliste weitere E-Mail-Adressen hinzuzufügen, wählen Sie Absender hinzufügen.

   c. (Optional) Wählen Sie unter Empfänger den Empfänger aus.

      Um Ihrer Empfängerliste weitere Benutzer hinzuzufügen, wählen Sie Neuen Benutzer hinzufügen. Sie müssen Ihrem IAM Identity Center-Shop neue Benutzer hinzufügen, bevor Sie sie zu Ihrem Alarmmodell hinzufügen können. Weitere Informationen finden Sie unter Empfänger verwalten.

   d. (Optional) Geben Sie unter Zusätzliche benutzerdefinierte Nachricht eine Nachricht ein, die beschreibt, was der Alarm erkennt und welche Maßnahmen die Empfänger ergreifen sollten.

8. Im Abschnitt Instanz können Sie alle Alarminstanzen aktivieren oder deaktivieren, die auf der Grundlage dieses Alarmmodells erstellt wurden.
9. Gehen Sie im Abschnitt Erweiterte Einstellungen wie folgt vor:

   a. Für den Acknowledge-Flow können Sie Benachrichtigungen aktivieren oder deaktivieren.

      - Wenn Sie Aktiviert wählen, erhalten Sie eine Benachrichtigung, wenn sich der Alarmstatus ändert. Sie müssen die Benachrichtigung bestätigen, bevor der Alarmstatus wieder normal werden kann.
      - Wenn Sie Deaktiviert wählen, ist keine Aktion erforderlich. Der Alarm wechselt automatisch in den Normalzustand, wenn die Messung in den angegebenen Bereich zurückkehrt.

      Weitere Informationen finden Sie unter Bestätigen Sie den Ablauf.

   b. Wählen Sie für Berechtigungen eine der folgenden Optionen aus:

      - Sie können eine neue Rolle anhand von AWS Richtlinienvorlagen erstellen und AWS IoT Events automatisch eine IAM-Rolle für Sie erstellen.
      - Sie können eine vorhandene IAM-Rolle verwenden, die es diesem Alarmmodell ermöglicht, Aktionen auszuführen und auf andere AWS Ressourcen zuzugreifen.

      Weitere Informationen finden Sie unter Identitäts- und Zugriffsverwaltung für AWS IoT Events.

   c. Für zusätzliche Benachrichtigungseinstellungen können Sie Ihre AWS Lambda Funktion zur Verwaltung von Alarmbenachrichtigungen bearbeiten. Wählen Sie eine der folgenden Optionen für Ihre AWS Lambda Funktion:

      - Neue AWS Lambda Funktion erstellen — AWS IoT Events erstellt eine neue AWS Lambda Funktion für Sie.
      - Eine bestehende AWS Lambda Funktion verwenden — Verwenden Sie eine bestehende AWS Lambda Funktion, indem Sie einen AWS Lambda Funktionsnamen wählen.

      Weitere Hinweise zu den möglichen Aktionen finden Sie unterArbeiten mit anderen - AWS Services.

d. (Optional) Für die Aktion Status festlegen können Sie eine oder mehrere AWS IoT Events Aktionen hinzufügen, die ausgeführt werden sollen, wenn sich der Alarmstatus ändert.

10. (Optional) Sie können Tags hinzufügen, um Ihre Alarme zu verwalten. Weitere Informationen finden Sie unter AWS IoT EventsRessourcen taggen.
11. Wählen Sie Erstellen aus.

Auf Alarme reagieren

Wenn Sie den Bestätigungsfluss aktiviert haben, erhalten Sie Benachrichtigungen, wenn sich der Alarmstatus ändert. Um auf den Alarm zu reagieren, können Sie den Alarm bestätigen, deaktivieren, aktivieren, zurücksetzen oder die Schlummerfunktion aktivieren.

Auf Alarme reagieren (Konsole)

Im Folgenden wird gezeigt, wie Sie auf einen Alarm in der AWS IoT Events Konsole reagieren.

1. Melden Sie sich an der AWS IoT Events-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Alarmmodelle aus.
3. Wählen Sie das Zielalarmmodell aus.
4. Wählen Sie im Bereich Liste der Alarme den Zielalarm aus.
5. Sie können unter Aktionen eine der folgenden Optionen wählen:

   - Bestätigen — Der Alarm wechselt in den `ACKNOWLEDGED` Status.
   - Deaktivieren — Der Alarm wechselt in den `DISABLED` Status.
   - Aktivieren — Der Alarm wechselt in den `NORMAL` Status.
   - Reset — Der Alarm wechselt in den `NORMAL` Status.
   - Schalten Sie die Schlummerfunktion ein und gehen Sie dann wie folgt vor:

     1. Wählen Sie die Schlummerlänge oder geben Sie eine benutzerdefinierte Schlummerlänge ein.
     2. Wählen Sie Speichern aus.

     Der Alarm wechselt in den Status `SNOOZE_DISABLED`

   Weitere Informationen zu den Alarmzuständen finden Sie unterBestätigen Sie den Ablauf.

Auf Alarme reagieren (API)

Um auf einen oder mehrere Alarme zu reagieren, können Sie die folgenden AWS IoT Events API-Operationen verwenden:

- BatchAcknowledgeAlarm
- BatchDisableAlarm
- BatchEnableAlarm
- BatchResetAlarm
- BatchSnoozeAlarm

Verwaltung von Alarmbenachrichtigungen

AWS IoT Eventsverwendet eine Lambda-Funktion zur Verwaltung von Alarmbenachrichtigungen. Sie können die von bereitgestellte Lambda-Funktion verwenden AWS IoT Events oder eine neue erstellen.

Erstellen einer Lambda-Funktion

AWS IoT Eventsbietet eine Lambda-Funktion, mit der Alarme E-Mail- und SMS-Benachrichtigungen senden und empfangen können.

Voraussetzungen

Die folgenden Anforderungen gelten, wenn Sie eine Lambda-Funktion für Alarme erstellen:

- Wenn Ihr Alarm E-Mail- oder SMS-Benachrichtigungen sendet, benötigen Sie eine IAM-Rolle, die die Arbeit mit Amazon SES und Amazon SNS ermöglichtAWS Lambda.

  Beispiel für eine Richtlinie:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ses:GetIdentityVerificationAttributes",
                "ses:SendEmail",
```

```
                "ses:VerifyEmailIdentity"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish",
                "sns:OptInPhoneNumber",
                "sns:CheckIfPhoneNumberIsOptedOut"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:*"
        }
    ]
}
```

- Sie müssen dieselbe AWS Region für AWS IoT Events sowohl als auch wählenAWS Lambda. Eine Liste der unterstützten Regionen finden Sie unter AWS IoT EventsEndpunkte und Kontingente und AWS LambdaEndpunkte und Kontingente in der. Allgemeine Amazon Web Services-Referenz

Bereitstellung einer Lambda-Funktion

In diesem Tutorial wird eine AWS CloudFormation Vorlage verwendet, um eine Lambda-Funktion bereitzustellen. Diese Vorlage erstellt automatisch eine IAM-Rolle, die es der Lambda-Funktion ermöglicht, mit Amazon SES und Amazon SNS zu arbeiten.

Im Folgenden wird gezeigt, wie Sie AWS Command Line Interface (AWS CLI) verwenden, um einen Stack zu erstellen. CloudFormation

1. Führen Sie im Terminal Ihres Geräts den Befehl aus, `aws --version` um zu überprüfen, ob Sie den installiert habenAWS CLI. Weitere Informationen finden Sie unter Installieren der AWS CLI im AWS Command Line Interface-Leitfaden.
2. Führen Sie den Vorgang aus`aws configure list`, um zu überprüfen, ob Sie das AWS CLI in der AWS Region konfiguriert haben, die alle Ihre AWS Ressourcen für dieses Tutorial enthält.

Weitere Informationen finden Sie unter Konfiguration von AWS CLI im AWS Command Line InterfaceBenutzerhandbuch

3. Laden Sie die CloudFormation Vorlage NotificationLambda.Template.Yaml.zip herunter.

> **Note**
>
> Wenn Sie Schwierigkeiten beim Herunterladen der Datei haben, finden Sie die Vorlage auch im. CloudFormation Vorlage

4. Entpacken Sie den Inhalt und speichern Sie die Datei lokal als `notificationLambda.template.yaml`.
5. Öffnen Sie ein Terminal auf Ihrem Gerät und navigieren Sie zu dem Verzeichnis, in das Sie die `notificationLambda.template.yaml` Datei heruntergeladen haben.
6. Führen Sie den folgenden Befehl aus, um einen CloudFormation Stack zu erstellen:

```
aws cloudformation create-stack --stack-name notificationLambda-stack --template-
body file://notificationLambda.template.yaml --capabilities CAPABILITY_IAM
```

Sie können diese CloudFormation Vorlage ändern, um die Lambda-Funktion und ihr Verhalten anzupassen.

> **Note**
>
> AWS Lambdawiederholt Funktionsfehler zweimal. Wenn die Kapazität der Funktion nicht für die Verarbeitung aller eingehenden Anforderungen ausreicht, verbleiben die an die Funktion zu sendenden Ereignisse möglicherweise stunden- oder tagelang in der Warteschlange. Sie können für die Funktion eine Warteschlange für unzugestellte Nachrichten (DLQ) konfigurieren, um Ereignisse aufzuzeichnen, die nicht erfolgreich verarbeitet wurden. Weitere Informationen finden Sie unter Asynchroner Aufruf im AWS Lambda Entwicklerhandbuch.

Sie können den Stack auch in der Konsole erstellen oder konfigurieren. CloudFormation Weitere Informationen finden Sie im AWS CloudFormationBenutzerhandbuch unter Arbeiten mit Stacks.

Eine benutzerdefinierte Lambda-Funktion erstellen

Sie können eine Lambda-Funktion erstellen oder die von AWS IoT Events bereitgestellte ändern.

Die folgenden Anforderungen gelten, wenn Sie eine benutzerdefinierte Lambda-Funktion erstellen.

- Fügen Sie Berechtigungen hinzu, die es Ihrer Lambda-Funktion ermöglichen, bestimmte Aktionen auszuführen und auf AWS Ressourcen zuzugreifen.
- Wenn Sie die von bereitgestellte Lambda-Funktion verwendenAWS IoT Events, stellen Sie sicher, dass Sie die Python 3.7-Laufzeit wählen.

Beispiel für eine Lambda-Funktion:

```
import boto3
import json
import logging
import datetime
logger = logging.getLogger()
logger.setLevel(logging.INFO)
ses = boto3.client('ses')
sns = boto3.client('sns')
def check_value(target):
  if target:
    return True
  return False

# Check whether email is verified. Only verified emails are allowed to send emails to
 or from.
def check_email(email):
  if not check_value(email):
    return False
  result = ses.get_identity_verification_attributes(Identities=[email])
  attr = result['VerificationAttributes']
  if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
      logging.info('Verification email for {} sent. You must have all the emails
 verified before sending email.'.format(email))
      ses.verify_email_identity(EmailAddress=email)
      return False
  return True

# Check whether the phone holder has opted out of receiving SMS messages from your
 account
def check_phone_number(phone_number):
  try:
    result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
    if (result['isOptedOut']):
```

```
        logger.info('phoneNumber {} is not opt in of receiving SMS messages. Phone
 number must be opt in first.'.format(phone_number))
        return False
    return True
  except Exception as e:
    logging.error('Your phone number {} must be in E.164 format in SSO. Exception
 thrown: {}'.format(phone_number, e))
    return False

def check_emails(emails):
  result = True
  for email in emails:
      if not check_email(email):
          result = False
  return result

def lambda_handler(event, context):
  logging.info('Received event: ' + json.dumps(event))
  nep = json.loads(event.get('notificationEventPayload'))
  alarm_state = nep['alarmState']
  default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
  timestamp =
 datetime.datetime.utcfromtimestamp(float(nep['stateUpdateTime'])/1000).strftime('%Y-
%m-%d %H:%M:%S')
  alarm_msg = "{} {} {} at {} UTC ".format(nep['alarmModelName'], nep.get('keyValue',
 'Singleton'), alarm_state['stateName'], timestamp)
  default_msg += 'Sev: ' + str(nep['severity']) + '\n'
  if (alarm_state['ruleEvaluation']):
    property = alarm_state['ruleEvaluation']['simpleRule']['inputProperty']
    default_msg += 'Current Value: ' + str(property) + '\n'
    operator = alarm_state['ruleEvaluation']['simpleRule']['operator']
    threshold = alarm_state['ruleEvaluation']['simpleRule']['threshold']
    alarm_msg += '({} {} {})'.format(str(property), operator, str(threshold))
  default_msg += alarm_msg + '\n'

  emails = event.get('emailConfigurations', [])
  logger.info('Start Sending Emails')
  for email in emails:
    from_adr = email.get('from')
    to_adrs = email.get('to', [])
    cc_adrs = email.get('cc', [])
    bcc_adrs = email.get('bcc', [])
    msg = default_msg + '\n' + email.get('additionalMessage', '')
    subject = email.get('subject', alarm_msg)
```

```
    fa_ver = check_email(from_adr)
    tas_ver = check_emails(to_adrs)
    ccas_ver = check_emails(cc_adrs)
    bccas_ver = check_emails(bcc_adrs)
    if (fa_ver and tas_ver and ccas_ver and bccas_ver):
      ses.send_email(Source=from_adr,
                     Destination={'ToAddresses': to_adrs, 'CcAddresses': cc_adrs,
'BccAddresses': bcc_adrs},
                     Message={'Subject': {'Data': subject}, 'Body': {'Text': {'Data':
msg}}})
      logger.info('Emails have been sent')

 logger.info('Start Sending SNS message to SMS')
 sns_configs = event.get('smsConfigurations', [])
 for sns_config in sns_configs:
   sns_msg = default_msg + '\n' + sns_config.get('additionalMessage', '')
   phone_numbers = sns_config.get('phoneNumbers', [])
   sender_id = sns_config.get('senderId')
   for phone_number in phone_numbers:
       if check_phone_number(phone_number):
         if check_value(sender_id):
           sns.publish(PhoneNumber=phone_number, Message=sns_msg,
MessageAttributes={'AWS.SNS.SMS.SenderID':{'DataType': 'String','StringValue':
sender_id}})
         else:
           sns.publish(PhoneNumber=phone_number, Message=sns_msg)
         logger.info('SNS messages have been sent')
```

Weitere Informationen finden Sie unter Was ist AWS Lambda? im AWS Lambda-Entwicklerhandbuch.

CloudFormation Vorlage

Verwenden Sie die folgende CloudFormation Vorlage, um Ihre Lambda-Funktion zu erstellen.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Notification Lambda for Alarm Model'
Resources:
  NotificationLambdaRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
```

```
            Service: lambda.amazonaws.com
          Action: sts:AssumeRole
    Path: "/"
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/AWSLambdaExecute'
    Policies:
      - PolicyName: "NotificationLambda"
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action:
                - "ses:GetIdentityVerificationAttributes"
                - "ses:SendEmail"
                - "ses:VerifyEmailIdentity"
              Resource: "*"
            - Effect: "Allow"
              Action:
                - "sns:Publish"
                - "sns:OptInPhoneNumber"
                - "sns:CheckIfPhoneNumberIsOptedOut"
              Resource: "*"
            - Effect: "Deny"
              Action:
                - "sns:Publish"
              Resource: "arn:aws:sns:*:*:*"
NotificationLambdaFunction:
  Type: AWS::Lambda::Function
  Properties:
    Role: !GetAtt NotificationLambdaRole.Arn
    Runtime: python3.7
    Handler: index.lambda_handler
    Timeout: 300
    MemorySize: 3008
    Code:
      ZipFile: |
        import boto3
        import json
        import logging
        import datetime
        logger = logging.getLogger()
        logger.setLevel(logging.INFO)
        ses = boto3.client('ses')
        sns = boto3.client('sns')
```

```
        def check_value(target):
          if target:
            return True
          return False

        # Check whether email is verified. Only verified emails are allowed to send
emails to or from.
        def check_email(email):
          if not check_value(email):
            return False
          result = ses.get_identity_verification_attributes(Identities=[email])
          attr = result['VerificationAttributes']
          if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
              logging.info('Verification email for {} sent. You must have all the
emails verified before sending email.'.format(email))
              ses.verify_email_identity(EmailAddress=email)
              return False
          return True

        # Check whether the phone holder has opted out of receiving SMS messages from
your account
        def check_phone_number(phone_number):
          try:
            result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
            if (result['isOptedOut']):
                logger.info('phoneNumber {} is not opt in of receiving SMS messages.
Phone number must be opt in first.'.format(phone_number))
                return False
            return True
          except Exception as e:
            logging.error('Your phone number {} must be in E.164 format in SSO.
Exception thrown: {}'.format(phone_number, e))
            return False

        def check_emails(emails):
          result = True
          for email in emails:
              if not check_email(email):
                  result = False
          return result

        def lambda_handler(event, context):
          logging.info('Received event: ' + json.dumps(event))
          nep = json.loads(event.get('notificationEventPayload'))
```

```
            alarm_state = nep['alarmState']
            default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
            timestamp =
 datetime.datetime.utcfromtimestamp(float(nep['stateUpdateTime'])/1000).strftime('%Y-
%m-%d %H:%M:%S')
            alarm_msg = "{} {} {} at {} UTC ".format(nep['alarmModelName'],
 nep.get('keyValue', 'Singleton'), alarm_state['stateName'], timestamp)
            default_msg += 'Sev: ' + str(nep['severity']) + '\n'
            if (alarm_state['ruleEvaluation']):
              property = alarm_state['ruleEvaluation']['simpleRule']['inputProperty']
              default_msg += 'Current Value: ' + str(property) + '\n'
              operator = alarm_state['ruleEvaluation']['simpleRule']['operator']
              threshold = alarm_state['ruleEvaluation']['simpleRule']['threshold']
              alarm_msg += '({} {} {})'.format(str(property), operator, str(threshold))
            default_msg += alarm_msg + '\n'

            emails = event.get('emailConfigurations', [])
            logger.info('Start Sending Emails')
            for email in emails:
              from_adr = email.get('from')
              to_adrs = email.get('to', [])
              cc_adrs = email.get('cc', [])
              bcc_adrs = email.get('bcc', [])
              msg = default_msg + '\n' + email.get('additionalMessage', '')
              subject = email.get('subject', alarm_msg)
              fa_ver = check_email(from_adr)
              tas_ver = check_emails(to_adrs)
              ccas_ver = check_emails(cc_adrs)
              bccas_ver = check_emails(bcc_adrs)
              if (fa_ver and tas_ver and ccas_ver and bccas_ver):
                ses.send_email(Source=from_adr,
                               Destination={'ToAddresses': to_adrs, 'CcAddresses':
 cc_adrs, 'BccAddresses': bcc_adrs},
                               Message={'Subject': {'Data': subject}, 'Body': {'Text':
 {'Data': msg}}})
                logger.info('Emails have been sent')

            logger.info('Start Sending SNS message to SMS')
            sns_configs = event.get('smsConfigurations', [])
            for sns_config in sns_configs:
              sns_msg = default_msg + '\n' + sns_config.get('additionalMessage', '')
              phone_numbers = sns_config.get('phoneNumbers', [])
              sender_id = sns_config.get('senderId')
              for phone_number in phone_numbers:
```

```
                if check_phone_number(phone_number):
                  if check_value(sender_id):
                    sns.publish(PhoneNumber=phone_number, Message=sns_msg,
MessageAttributes={'AWS.SNS.SMS.SenderID':{'DataType': 'String','StringValue':
sender_id}})
                  else:
                    sns.publish(PhoneNumber=phone_number, Message=sns_msg)
                  logger.info('SNS messages have been sent')
```

Verwenden der Lambda-Funktion von AWS IoT Events

Die folgenden Anforderungen gelten, wenn Sie die Lambda-Funktion von verwendenAWS IoT Events, um Ihre Alarmbenachrichtigungen zu verwalten:

- Sie müssen die E-Mail-Adresse, mit der die E-Mail-Benachrichtigungen gesendet werden, in Amazon Simple Email Service (Amazon SES) verifizieren. Weitere Informationen finden Sie unter Verifizieren von E-Mail-Adressen in Amazon SES im Amazon Simple Email Service Developer Guide.

  Wenn Sie einen Bestätigungslink erhalten, klicken Sie auf den Link, um Ihre E-Mail-Adresse zu verifizieren. Sie können auch in Ihrem Spam-Ordner nach einer Bestätigungs-E-Mail suchen.
- Wenn Ihr Alarm SMS-Benachrichtigungen sendet, müssen Sie die internationale E.164-Formatierung für Telefonnummern verwenden. Dieses Format enthält+`<country-calling-code><area-code><phone-number>`.

  Beispiele für Telefonnummern:

Land	Lokale Telefonnummer	E.164 formatierte Nummer
Vereinigte Staaten	206-555-0100	+12065550100
Großbritannien und Nordirland	020-1234-1234	+442012341234
Litauen	+601+12345	+37060112345

  Um eine Landesvorwahl zu finden, gehen Sie zu countrycode.org.

Die von bereitgestellte Lambda-Funktion AWS IoT Events überprüft, ob Sie E.164-formatierte Telefonnummern verwenden. Die Telefonnummern werden jedoch nicht überprüft. Wenn Sie sicherstellen, dass Sie korrekte Telefonnummern eingegeben, aber keine SMS-Benachrichtigungen erhalten haben, können Sie sich an die Telefonanbieter wenden. Die Mobilfunkanbieter können die Nachrichten blockieren.

Empfänger verwalten

AWS IoT Eventsverwendet AWS IAM Identity Center (IAM Identity Center), um den SSO-Zugriff von Alarmempfängern zu verwalten. Damit der Alarm Benachrichtigungen an die Empfänger senden kann, müssen Sie IAM Identity Center aktivieren und Empfänger zu Ihrem IAM Identity Center-Shop hinzufügen. Weitere Informationen finden Sie im AWS IAM Identity CenterBenutzerhandbuch unter Benutzer hinzufügen.

Important

- Sie müssen dieselbe AWS Region für AWS IoT EventsAWS Lambda, und IAM Identity Center wählen.
- AWSOrganizations unterstützen jeweils nur eine IAM Identity Center-Region. Wenn Sie IAM Identity Center in einer anderen Region verfügbar machen möchten, müssen Sie zuerst Ihre aktuelle IAM Identity Center-Konfiguration löschen. Weitere Informationen finden Sie unter IAM Identity Center-Regionaldaten im AWS IAM Identity Center Benutzerhandbuch.

Sicherheit in AWS IoT Events

Die Sicherheit in der Cloud hat bei AWS höchste Priorität. Als AWS-Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat.

Sicherheit gilt zwischen AWS und Ihnen eine geteilte Verantwortung. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS-Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS-Compliance-Programms getestet und überprüft. Informationen zu den Compliance-Programmen, die für AWS IoT Events gelten, finden Sie unter Vom Compliance-Programm abgedeckte AWS-Services.
- Sicherheit in der Cloud – Ihr Verantwortungsumfang wird durch den AWS-Service bestimmt, den Sie verwenden. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation beschreibt, wie Sie das Modell der übergreifenden Verantwortlichkeit bei der Verwendung von AWS IoT Events anwenden können. Die folgenden Themen veranschaulichen, wie Sie AWS IoT Events zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer AWS IoT Events Ressourcen helfen können.

Themen

Identity and Access Management für AWS IoT Events

AWS Identity and Access Management (IAM) ist ein AWS-Service, mit dem ein Administrator den Zugriff auf AWS-Ressourcen sicher steuern kann. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (Berechtigungen besitzt) ist, um AWS IoT Events Ressourcen zu nutzen. IAM ist ein AWS-Service, den Sie ohne zusätzliche Kosten verwenden können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in AWS IoT Events.

Service-Benutzer: Wenn Sie den AWS IoT Events-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie für Ihre Arbeit weitere AWS IoT Events-Funktionen ausführen, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Unter Fehlerbehebung für AWS IoT Events-Identität und -Zugriff finden Sie nützliche Informationen für den Fall, dass Sie keinen Zugriff auf eine Funktion in AWS IoT Events haben.

Service-Administrator: Wenn Sie in Ihrem Unternehmen für AWS IoT Events-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf AWS IoT Events. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS IoT Events-Funktionen und Ressourcen Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die

Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit AWS IoT Events verwenden kann, finden Sie unter Featuresweise von AWS IoT Events mit IAM.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS IoT Events verfassen können. Beispiele für identitätsbasierte AWS IoT Events-Richtlinien, die Sie in IAM verwenden können, finden Sie unter AWS IoT EventsBeispiele für identitätsbasierte -Richtlinien.

Authentifizierung mit Identitäten

Authentifizierung ist die Art, wie Sie sich mit Ihren Anmeldeinformationen bei AWS anmelden. Die Authentifizierung (Anmeldung bei AWS) muss als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle erfolgen.

Sie können sich bei AWS als Verbundidentität mit Anmeldeinformationen anmelden, die über eine Identitätsquelle bereitgestellt werden. Benutzer von AWS IAM Identity Center (IAM Identity Center), die Single-Sign-on-Authentifizierung Ihres Unternehmens und Anmeldeinformationen für Google oder Facebook sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie auf AWS mithilfe des Verbunds zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der AWS Management Console oder beim AWS-Zugriffsportal anmelden. Weitere Informationen zum Anmelden bei AWS finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch von AWS-Anmeldung.

Bei programmgesteuertem Zugriff auf AWS bietet AWS ein Software Development Kit (SDK) und eine Command Line Interface (CLI, Befehlszeilenschnittstelle) zum kryptographischen Signieren Ihrer Anfragen mit Ihren Anmeldeinformationen. Wenn Sie keine AWS-Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenen Signieren von Anforderungen finden Sie unter Signieren von AWS-API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS empfiehlt beispielsweise die Verwendung von Multi-Faktor Authentifizierung (MFA), um die Sicherheit Ihres Kontos zu verbessern. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center-Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto-Root-Benutzer

Wenn Sie ein AWS-Konto neu erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen des Kontos hat. Diese Identität wird als AWS-Konto-Root-Benutzer bezeichnet. Für den Zugriff auf den Root-Benutzer müssen Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, die zur Erstellung des Kontos verwendet wurden. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder eine einzelne Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit spezifischen Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können

vorübergehend eine IAM-Rolle in der AWS Management Console übernehmen, indem Sie Rollen wechseln. Sie können eine Rolle annehmen, indem Sie eine AWS CLI oder AWS-API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff: Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center-Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen: Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. In einigen AWS-Services können Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff: Einige AWS-Services verwenden Features in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward access sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu

stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Serviceverknüpfte Rolle: Eine serviceverknüpfte Rolle ist ein Typ von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverbundene Rollen anzeigen, aber nicht bearbeiten.

- Anwendungen in Amazon EC2: Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und AWS CLI- oder AWS-API-Anforderungen durchführen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Erstellen Sie ein Instance-Profil, das an die Instance angefügt ist, um eine AWS-Rolle einer EC2-Instance zuzuweisen und die Rolle für sämtliche Anwendungen der Instance bereitzustellen. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Für die Zugriffssteuerung in AWS erstellen Sie Richtlinien und weisen diese den AWS-Identitäten oder -Ressourcen zu. Eine Richtlinie ist ein Objekt in AWS, das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und

Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Benutzerinformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem AWS-Konto anfügen können. Verwaltete Richtlinien umfassen von AWS verwaltete und von Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen:Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Dienst für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. SCPs schränken Berechtigungen für Entitäten in Mitgliedskonten einschließlich des jeweiligen Root-Benutzer des AWS-Kontos ein. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations-Benutzerhandbuch.
- Sitzungsrichtlinien:Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

Weitere Informationen

Weitere Informationen über Identitäts- und Zugriffsverwaltung für AWS IoT Events finden Sie auf den folgenden Seiten:

Featuresweise von AWS IoT Events mit IAM

Bevor Sie IAM zum Verwalten des Zugriffs auf AWS IoT Events verwenden, sollten Sie wissen, welche IAM-Funktionen für die Verwendung mit AWS IoT Events verfügbar sind. Einen Überblick über das Zusammenwirken von AWS IoT Events und anderen AWS-Services mit IAM finden Sie unter AWS-Services, die mit IAM funktionieren im IAM-Benutzerhandbuch.

Themen

Identitätsbasierte AWS IoT Events-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen erteilt oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. AWS IoT Events unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Das Element `Action` einer identitätsbasierten IAM-Richtlinie beschreibt die spezifischen Aktionen, die von der Richtlinie zugelassen oder abgelehnt werden. Richtlinienaktionen haben normalerweise denselben Namen wie die zugehörige AWS-API-Operation. Die Aktion wird in einer Richtlinie verwendet, um Berechtigungen zur Durchführung der zugehörigen Aktion zu gewähren.

Richtlinienaktionen in AWS IoT Events verwenden das folgende Präfix vor der Aktion: `iotevents:`. Um beispielsweise jemandem die Berechtigung zum Erstellen einer AWS IoT Events Eingabe mit der AWS IoT Events `CreateInput` API-Operation zu erteilen, fügen Sie die `iotevents:CreateInput` Aktion in seine Richtlinie ein. Um jemandem die Berechtigung zum Senden einer Eingabe mit

der AWS IoT Events `BatchPutMessage` -API-Operation zu erteilen, fügen Sie die `iotevents-data:BatchPutMessage` Aktion in seine Richtlinie ein. Richtlinienanweisungen müssen ein `Action`- oder `NotAction`-Element enthalten. AWS IoT Events definiert seinen eigenen Satz an Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service durchführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

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

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "iotevents:Describe*"
```

Eine Liste der AWS IoT Events-Aktionen finden Sie im IAM-Benutzerhandbuch unter Aktionen, die von AWS IoT Eventsdefiniert werden.

Ressourcen

Das Element `Resource` gibt die Objekte an, auf die die Aktion angewendet wird. Anweisungen müssen entweder ein `Resource`- oder ein `NotResource`-Element enthalten. Sie geben eine Ressource unter Verwendung eines ARN oder eines Platzhalters (*) an, um anzugeben, dass die Anweisung für alle Ressourcen gilt.

Die AWS IoT EventsDetektormodellressource hat den folgenden ARN:

```
arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/${detectorModelName}
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon-Ressourcennamen (ARNs) und AWS-Service-Namespaces.

Um beispielsweise das `Foobar`Detektormodell in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN:

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/Foobar"
```

Um alle Instances anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/*"
```

Einige AWS IoT Events-Aktionen, z. B. zum Erstellen von Ressourcen, können auf bestimmten Ressourcen nicht ausgeführt werden. In diesen Fällen müssen Sie den Platzhalter (*) verwenden.

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

Bei einigen AWS IoT Events-API-Aktionen sind mehrere Ressourcen beteiligt. Beispielsweise `CreateDetectorModel` verweist auf Eingaben in seinen Bedingungsanweisungen, sodass ein Benutzer über Berechtigungen zur Verwendung der Eingabe und des Detektormodells verfügen muss. Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, trennen Sie die ARNs durch Kommata voneinander.

```
"Resource": [
      "resource1",
      "resource2"
```

Eine Liste der AWS IoT Events-Ressourcentypen und ihrer ARNs finden Sie unter Von AWS IoT Events definierte Ressourcentypen im IAM-Benutzerhandbuch. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von AWS IoT Events definierte Aktionen.

Bedingungsschlüssel

Mithilfe des Elements `Condition`(oder des Blocks `Condition`) können Sie die Bedingungen angeben, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungs-Operatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die

Bedingung mittels einer logischen OR-Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags (Markierungen) im IAM-Benutzerhandbuch.

AWS IoT Events stellt keine servicespezifischen Bedingungsschlüssel bereit, unterstützt jedoch die Verwendung einiger globaler Bedingungsschlüssel. Informationen zum Anzeigen aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.“

Beispiele

Beispiele für identitätsbasierte AWS IoT Events-Richtlinien finden Sie unter AWS IoT EventsBeispiele für identitätsbasierte -Richtlinien.

Ressourcenbasierte AWS IoT Events-Richtlinien

AWS IoT Events unterstützt ressourcenbasierte Richtlinien nicht.“ Ein Beispiel für eine detaillierte Seite zu ressourcenbasierten Richtlinien finden Sie unter https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Autorisierung auf der Basis von AWS IoT Events-Tags

Sie können Tags an AWS IoT Events-Ressourcen anfügen oder Tags in einer Anforderung an AWS IoT Events übergeben. Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `iotevents:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden. Weitere Informationen über das Markieren von AWS IoT Events-Ressourcen mit Tags finden Sie unter Markieren Ihrer AWS IoT Events-Ressourcen.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Anzeigen von AWS IoT Events*Eingaben* basierend auf Tags.

AWS IoT EventsIAM-Rollen

Eine IAM-Rolle ist eine Entität in Ihrem AWS-Konto mit spezifischen Berechtigungen.

Verwenden temporärer Anmeldeinformationen mit AWS IoT Events

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Sie erhalten temporäre Sicherheitsanmeldeinformationen, indem Sie AWS Security Token Service (AWS STS)-API-Operationen wie AssumeRole oder aufrufenGetFederationToken.

AWS IoT Events unterstützt nicht die Verwendung temporärer Anmeldeinformationen.

Serviceverknüpfte Rollen

Serviceverknüpfte Rollen erlauben AWS-Services den Zugriff auf Ressourcen in anderen Services, um eine Aktion in Ihrem Auftrag auszuführen. Serviceverknüpfte Rollen werden in Ihrem IAM-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.

AWS IoT Events unterstützt serviceverknüpfte Rollen nicht.

Servicerollen

Dieses Feature ermöglicht einem Service das Annehmen einer Servicerolle in Ihrem Namen. Diese Rolle gewährt dem Service Zugriff auf Ressourcen in anderen Diensten, um eine Aktion in Ihrem Namen auszuführen. Servicerollen werden in Ihrem IAM-Konto angezeigt und gehören zum Konto. Dies bedeutet, dass ein IAM-Administrator die Berechtigungen für diese Rolle ändern kann. Dies kann jedoch die Funktion des Services beeinträchtigen.

AWS IoT Events unterstützt Servicerollen.

AWS IoT EventsBeispiele für identitätsbasierte -Richtlinien

Benutzer und Rollen haben standardmäßig nicht die Berechtigung, AWS IoT Events-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben ausführen, die die AWS Management Console-, AWS CLI- oder AWS-API benutzen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den -Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien sind sehr leistungsfähig. Sie können festlegen, ob jemand AWS IoT Events-Ressourcen in Ihrem Konto erstellen oder löschen oder auf sie zugreifen kann.. Dies kann zusätzliche Kosten für Ihr AWS-Konto verursachen. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte beim Verwendung von AWS verwalteter Richtlinien – Um schnell mit der Verwendung von AWS IoT Events zu beginnen, verwenden Sie von AWS verwaltete Richtlinien, um Ihren Mitarbeitern die von ihnen benötigten Berechtigungen zu gewähren. Diese Richtlinien sind bereits in Ihrem Konto verfügbar und werden von AWS gewartet und aktualisiert. Weitere Informationen finden Sie unter Erste Schritte zur Verwendung von Berechtigungen mit AWS-verwalteten Richtlinien im IAM-Benutzerhandbuch.
- Gewähren Sie die geringstmöglichen Berechtigungen – Gewähren Sie beim Erstellen benutzerdefinierter Richtlinien nur die Berechtigungen, die zum Ausführen einer Aufgabe erforderlich sind. Beginnen Sie mit einem Mindestsatz von Berechtigungen und gewähren Sie zusätzliche Berechtigungen wie erforderlich. Dies ist sicherer, als mit Berechtigungen zu beginnen, die zu weit gefasst sind, und dann später zu versuchen, sie zu begrenzen. Weitere Informationen finden Sie unter Gewähren von geringsten Rechten im IAM-Benutzerhandbuch.
- MFA für sensible Vorgänge aktivieren – Fordern Sie Benutzer für zusätzliche Sicherheit auf, Multi-Faktor-Authentifizierung (MFA) zu verwenden, um auf sensible Ressourcen oder API-Operationen zuzugreifen. Weitere Informationen finden Sie unter Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.
- Verwenden Sie Richtlinienbedingungen, um zusätzliche Sicherheit zu bieten – Definieren Sie die Bedingungen, unter denen Ihre identitätsbasierten Richtlinien den Zugriff auf eine Ressource zulassen, soweit praktikabel. Beispielsweise können Sie Bedingungen schreiben, die eine Reihe von zulässigen IP-Adressen festlegen, von denen eine Anforderung stammen muss. Sie können auch Bedingungen schreiben, die Anforderungen nur innerhalb eines bestimmten Datums- oder

Zeitbereichs zulassen oder die Verwendung von SSL oder MFA fordern. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

Verwenden der AWS IoT Events-Konsole

Um auf die AWS IoT Events-Konsole zuzugreifen, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen Ihnen das Auflisten und Anzeigen von Details zu den AWS IoT Events-Ressourcen in Ihrem AWS-Konto gestatten. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Um sicherzustellen, dass diese Entitäten dennoch die AWS IoT Events-Konsole verwenden können, fügen Sie den Entitäten auch die folgende von AWS verwaltete Richtlinie an. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotevents-data:BatchPutMessage",
                "iotevents-data:BatchUpdateDetector",
                "iotevents:CreateDetectorModel",
                "iotevents:CreateInput",
                "iotevents:DeleteDetectorModel",
                "iotevents:DeleteInput",
                "iotevents-data:DescribeDetector",
                "iotevents:DescribeDetectorModel",
                "iotevents:DescribeInput",
                "iotevents:DescribeLoggingOptions",
                "iotevents:ListDetectorModelVersions",
                "iotevents:ListDetectorModels",
                "iotevents-data:ListDetectors",
                "iotevents:ListInputs",
                "iotevents:ListTagsForResource",
                "iotevents:PutLoggingOptions",
                "iotevents:TagResource",
                "iotevents:UntagResource",
                "iotevents:UpdateDetectorModel",
```

```
                "iotevents:UpdateInput",
                "iotevents:UpdateInputRouting"
            ],
            "Resource": "arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/
${detectorModelName}",
            "Resource": "arn:${Partition}:iotevents:${Region}:${Account}:input/
${inputName}"
        }
    ]
}
```

Für Benutzer, die nur Aufrufe an die AWS CLI oder AWS-API durchführen, müssen Sie keine Mindestberechtigungen in der Konsole erteilen. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die den API-Operation entsprechen, die Sie ausführen möchten.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die -Benutzern die Berechtigung zum Anzeigen der Inline-Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen für die Ausführung dieser Aktion auf der Konsole oder für die programmgesteuerte Ausführung über die AWS CLI oder die 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": "*"
        }
    ]
}
```

Zugreifen auf eine AWS IoT Events Eingabe

In diesem Beispiel möchten Sie einem Benutzer in Ihrem AWS-Konto Zugriff auf eine Ihrer AWS IoT Events Eingaben gewähren, `exampleInput`. Sie möchten dem Benutzer auch erlauben, Eingaben hinzuzufügen, zu aktualisieren und zu löschen.

Die Richtlinie gewährt dem Benutzer die Berechtigungen `iotevents:ListInputs`, `iotevents:DescribeInput`, `iotevents:CreateInput`, `iotevents:DeleteInput` und `iotevents:UpdateInput`. Ein Beispiel für eine exemplarische exemplarische Vorgehensweise für Amazon Simple Storage Service (Amazon S3), die Benutzern Berechtigungen erteilt und diese mithilfe der Konsole testet, finden Sie unter Ein Beispiel-Walkthrough: Verwenden von Benutzerrichtlinien zur Steuerung des Zugriffs auf Ihren Bucket.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListInputsInConsole",
         "Effect":"Allow",
         "Action":[
            "iotevents:ListInputs"
         ],
         "Resource":"arn:aws:iotevents:::*"
      },
      {
         "Sid":"ViewSpecificInputInfo",
         "Effect":"Allow",
         "Action":[
```

```
            "iotevents:DescribeInput"
         ],
         "Resource":"arn:aws:iotevents:::exampleInput"
      },
      {
         "Sid":"ManageInputs",
         "Effect":"Allow",
         "Action":[
            "iotevents:CreateInput",
            "iotevents:DeleteInput",
            "iotevents:DescribeInput",
            "iotevents:ListInputs",
            "iotevents:UpdateInput"
         ],
         "Resource":"arn:aws:iotevents:::exampleInput/*"
      }
   ]
}
```

Anzeigen von AWS IoT Events*Eingaben* basierend auf Tags

Sie können in Ihrer identitätsbasierten Richtlinie Bedingungen für die Steuerung des Zugriffs auf AWS IoT Events-Ressourcen auf der Basis von Tags verwenden. Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen können, die das Anzeigen einer *Eingabe* erlaubt. Die Berechtigung wird jedoch nur erteilt, wenn das *Eingabe*-Tag den Wert des Benutzernamens dieses Benutzers `Owner` hat. Diese Richtlinie gewährt auch die Berechtigungen, die für die Ausführung dieser Aktion auf der Konsole erforderlich sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListInputsInConsole",
            "Effect": "Allow",
            "Action": "iotevents:ListInputs",
            "Resource": "*"
        },
        {
            "Sid": "ViewInputsIfOwner",
            "Effect": "Allow",
            "Action": "iotevents:ListInputs",
            "Resource": "arn:aws:iotevents:*:*:input/*",
```

```
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Sie können diese Richtlinie den -Benutzern in Ihrem Konto zuweisen. Wenn ein Benutzer mit dem Namen `richard-roe` versucht, eine AWS IoT Events *`Eingabe`* anzuzeigen, muss die *`Eingabe`* mit dem Tag `Owner=richard-roe` oder versehen sein`owner=richard-roe`. Andernfalls wird der Zugriff abgelehnt. Der Tag-Schlüssel `Owner` der Bedingung stimmt sowohl mit `Owner` als auch mit `owner` überein, da die Namen von Bedingungsschlüsseln nicht zwischen Groß- und Kleinschreibung unterscheiden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

Vermeidung des Problems des verwirrten Stellvertreters (dienstübergreifend)

> **Note**
>
> - Der AWS IoT Events Service ermöglicht es Kunden nur, Rollen zu verwenden, um Aktionen in demselben Konto zu starten, in dem eine Ressource erstellt wurde. Das bedeutet, dass ein Confused Deputy Attack mit diesem Service nicht durchgeführt werden kann.
> - Diese Seite dient Kunden als Referenz, um zu erfahren, wie das Problem mit dem verwirrten Stellvertreter funktioniert. Sie kann verhindert werden, wenn kontoübergreifende Ressourcen für den AWS IoT Events Service zugelassen wurden.

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine Entität, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine Entität mit größeren Rechten zwingen kann, die Aktion auszuführen. In AWS kann der dienstübergreifende Identitätswechsel zu Confused-Deputy-Problem führen. Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der aufrufende Service kann manipuliert werden, um seine Berechtigungen zu verwenden, um Aktionen auf die Ressourcen eines anderen Kunden auszuführen, für die er sonst keine Zugriffsberechtigung haben sollte. Um dies zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Wir empfehlen die Verwendung der globalen Bedingungskontext-Schlüssel `aws:SourceArn` und `aws:SourceAccount` in ressourcenbasierten Richtlinien, um die Berechtigungen, die AWS IoT Events einem anderen Service erteilt, auf eine bestimmte Ressource zu beschränken. Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie beide globale Bedingungskontextschlüssel verwenden, um Berechtigungen einzuschränken. Wenn Sie beide globale Bedingungskontextschlüssel verwenden und der `aws:SourceArn`-Wert die Konto-ID enthält, müssen der `aws:SourceAccount`-Wert und das Konto im `aws:SourceArn`-Wert dieselbe Konto-ID verwenden, wenn sie in der gleichen Richtlinienanweisung verwendet wird.

Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden. Der Wert von `aws:SourceArn` muss dem Meldermodell oder dem Alarmmodell entsprechen, das der `sts:AssumeRole` Anfrage zugeordnet ist.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontextschlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel mit Platzhaltern (`aws:SourceArn`) * für die unbekannten Teile des ARN. Zum Beispiel `arn:aws:iotevents:*:123456789012:*`.

Die folgenden Beispiele zeigen, wie Sie die Kontextschlüssel `aws:SourceArn` und die `aws:SourceAccount` globale Bedingung verwenden können, AWS IoT Events um das Problem des verwirrten Stellvertreters zu vermeiden.

Themen

- Beispiel 1: Zugriff auf ein Detektormodell
- Beispiel 2: Zugriff auf ein Alarmmodell
- Beispiel 3: Zugriff auf eine Ressource in einer bestimmten Region
- Beispiel 4: Protokollierungsoptionen

Beispiel 1: Zugriff auf ein Detektormodell

Die folgende Rolle kann nur für den Zugriff auf ein DetectorModel benanntes Objekt verwendet werden`foo`.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:region:account_id:detectorModel/foo"
        }
      }
    }
  ]
 }
}
```

Beispiel 2: Zugriff auf ein Alarmmodell

Die folgende Rolle kann nur für den Zugriff auf ein beliebiges Alarmmodell verwendet werden.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
```

```
          "aws:SourceArn": "arn:aws:iotevents:region:account_id:alarmModel/*"
        }
      }
    }
  ]
}
```

Beispiel 3: Zugriff auf eine Ressource in einer bestimmten Region

Das folgende Beispiel zeigt eine Rolle, mit der Sie auf eine Ressource in einer bestimmten Region zugreifen können. Die Region in diesem Beispiel ist *us-east-1*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-east-1:account_id:*"
        }
      }
    }
  ]
}
```

Beispiel 4: Protokollierungsoptionen

Um eine Rolle für Protokollierungsoptionen bereitzustellen, müssen Sie zulassen, dass sie für jede Ressource in IoT Events übernommen wird. Dementsprechend müssen Sie einen Platzhalter (*) für den Ressourcentyp und den Ressourcennamen verwenden.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:region:account_id:*"
        }
      }
    }
  ]
}
```

Fehlerbehebung für AWS IoT Events-Identität und -Zugriff

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die beim Arbeiten mit AWS IoT Events und IAM auftreten könnten.

Themen

- Ich bin nicht autorisiert, eine Aktion in AWS IoT Events auszuführen.
- Ich bin nicht zur Ausführung von iam:PassRole autorisiert.
- Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine AWS IoT Events-Ressourcen gewähren

Ich bin nicht autorisiert, eine Aktion in AWS IoT Events auszuführen.

Wenn die AWS Management Console Ihnen mitteilt, dass Sie nicht zur Ausführung einer Aktion autorisiert sind, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort bereitgestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zu verwenden, um Details zu einer *`Eingabe`* anzuzeigen, aber keine `iotevents:`*`ListInputs`* Berechtigungen besitzt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotevents:ListInputs on resource: my-example-input
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`my-example-input`* auf die Ressource `iotevents:`*`ListInput`* zugreifen zu können.

Ich bin nicht zur Ausführung von **`iam:PassRole`** autorisiert.

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWS IoT Events übergeben zu können.

Einige AWS-Services erlauben die Übergabe einer vorhandenen Rolle an diesen Dienst, sodass keine neue Servicerolle oder serviceverknüpfte Rolle erstellt werden muss. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS IoT Events auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

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

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine AWS IoT Events-Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem

die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob AWS IoT Events diese Funktionen unterstützt, finden Sie unter Featuresweise von AWS IoT Events mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen für alle Ihre AWS-Konten finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen Ihrer AWS-Konto im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie AWS-Konten-Drittanbieter Zugriff auf Ihre Ressourcen bereitstellen, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von externen Benutzern im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Überwachung von AWS IoT Events

Die Überwachung ist ein wichtiger Teil der Aufrechterhaltung von Zuverlässigkeit, Verfügbarkeit und Performance von AWS IoT Events und Ihren AWS-Lösungen. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Ausfall an mehreren Stellen leichter debuggen können. Bevor Sie mit der Überwachung von AWS IoT Events beginnen, sollten Sie einen Überwachungsplan mit Antworten auf die folgenden Fragen erstellen:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen möchten Sie überwachen?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungs-Tools möchten Sie verwenden?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Im nächsten Schritt legen Sie einen Ausgangswert für die normale AWS IoT Events-Leistung in Ihrer Umgebung fest, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Speichern Sie bei der Überwachung von AWS IoT Events historische Überwachungsdaten, damit Sie diese mit aktuellen Leistungsdaten vergleichen, normale Leistungsmuster bestimmen, Leistungsprobleme erkennen und Methoden zur Fehlerbehebung ableiten können.

Wenn Sie beispielsweise Amazon EC2 verwenden, können Sie die CPU-Auslastung, Festplatten-I/O und Netzwerkauslastung für Ihre Instances überwachen. Wenn die Leistung außerhalb der festgelegten Grundwerte liegt, müssen Sie die Instance neu konfigurieren oder optimieren, um die CPU-Nutzung zu verringern, die Festplatten-I/O zu verbessern oder den Netzwerkverkehr zu reduzieren.

Themen

Überwachungstools

AWS bietet verschiedene Tools, mit deren Hilfe Sie AWS IoT Events überwachen können. Sie können einige dieser Tools so konfigurieren, dass diese die Überwachung für Sie übernehmen, während bei anderen Tools ein manuelles Eingreifen nötig ist. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Automatisierte Überwachungstools

Sie können die folgenden automatisierten Tools zur Überwachung von AWS IoT Events verwenden und möglicherweise auftretende Probleme melden:

- Amazon CloudWatch Logs — Überwachen Sie Ihre Protokolldateien AWS CloudTrail oder andere Quellen, speichern Sie sie und greifen Sie darauf zu. Weitere Informationen finden Sie unter Überwachung von Protokolldateien im CloudWatch Amazon-Benutzerhandbuch.
- CloudWatch Amazon-Ereignisse — Ordnen Sie Ereignisse zu und leiten Sie sie an eine oder mehrere Zielfunktionen oder -streams weiter, um Änderungen vorzunehmen, Statusinformationen zu erfassen und Korrekturmaßnahmen zu ergreifen. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch Events im CloudWatch Amazon-Benutzerhandbuch.

- AWS CloudTrailProtokollüberwachung — Teilen Sie Protokolldateien zwischen Konten, überwachen CloudTrail Sie Protokolldateien in Echtzeit, indem Sie sie an CloudWatch Logs senden, schreiben Sie Protokollverarbeitungsanwendungen in Java und stellen Sie sicher, dass sich Ihre Protokolldateien nach der Lieferung von nicht geändert haben. CloudTrail Weitere Informationen finden Sie unter Arbeiten mit CloudTrail Protokolldateien im AWS CloudTrailBenutzerhandbuch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Überwachung AWS IoT Events umfasst die manuelle Überwachung der Elemente, die von den CloudWatch Alarmen nicht abgedeckt werden. Die Konsolen-Dashboards AWS IoT Events CloudWatch, und andere AWS Konsolen-Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer AWS Umgebung. Zudem empfehlen wir die Überprüfung der Protokolldateien auf AWS IoT Events.

- Die AWS IoT Events-Konsole zeigt Folgendes:
  - Detektormodelle
  - Detektoren
  - Eingaben
  - Einstellungen
- Auf der CloudWatch Startseite wird Folgendes angezeigt:
  - Aktuelle Alarme und Status
  - Diagramme mit Alarmen und Ressourcen
  - Servicestatus

  Darüber hinaus können CloudWatch Sie Folgendes verwenden:
  - Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
  - Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
  - Durchsuchen und Suchen aller AWS-Ressourcenmetriken
  - Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

Überwachung mit Amazon CloudWatch

Wenn Sie ein AWS IoT Events Detektormodell entwickeln oder debuggen, müssen Sie wissen, was AWS IoT Events gerade passiert und welche Fehler dabei auftreten. Amazon CloudWatch überwacht

Ihre Amazon Web Services (AWS) -Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Damit CloudWatch erhalten Sie systemweiten Einblick in die Ressourcennutzung, die Anwendungsleistung und den Betriebszustand. Aktivieren Sie die CloudWatch Amazon-Protokollierung bei der Entwicklung von AWS IoT Events Detektormodellenenthält Informationen zur Aktivierung der CloudWatch Protokollierung fürAWS IoT Events. Um Protokolle wie das unten gezeigte zu generieren, müssen Sie den Ausführlichkeitsgrad auf „Debug“ setzen und ein oder mehrere Debug-Ziele angeben, d. h. einen Modellnamen des Detektors und ein optionales. KeyValue

Das folgende Beispiel zeigt einen Protokolleintrag auf CloudWatch DEBUG-Ebene, der von generiert wurde. AWS IoT Events


```
{
  "timestamp": "2019-03-15T15:56:29.412Z",
  "level": "DEBUG",
  "logMessage": "Summary of message evaluation",
  "context": "MessageEvaluation",
  "status": "Success",
  "messageId": "SensorAggregate_2th846h",
  "keyValue": "boiler_1",
  "detectorModelName": "BoilerAlarmDetector",
  "initialState": "high_temp_alarm",
  "initialVariables": {
    "high_temp_count": 1,
    "high_pressure_count": 1
  },
  "finalState": "no_alarm",
  "finalVariables": {
    "high_temp_count": 0,
    "high_pressure_count": 0
  },
  "message": "{ \"temp\": 34.9, \"pressure\": 84.5}",
  "messageType": "CUSTOMER_MESSAGE",
  "conditionEvaluationResults": [
    {
      "result": "True",
      "eventName": "alarm_cleared",
      "state": "high_temp_alarm",
      "lifeCycle": "OnInput",
      "hasTransition": true
    },
    {
      "result": "Skipped",
```

```
      "eventName": "alarm_escalated",
      "state": "high_temp_alarm",
      "lifeCycle": "OnInput",
      "hasTransition": true,
      "resultDetails": "Skipped due to transition from alarm_cleared event"
    },
    {
      "result": "True",
      "eventName": "should_recall_technician",
      "state": "no_alarm",
      "lifeCycle": "OnEnter",
      "hasTransition": true
    }
  ]
}
```

Protokollierung von AWS IoT Events-API-Aufrufen mit AWS CloudTrail

AWS IoT Eventsist in einen Dienst integriertAWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in AWS IoT Events ausgeführt wurden. CloudTrail erfasst alle API-Aufrufe AWS IoT Events als Ereignisse, einschließlich Aufrufe von der AWS IoT Events Konsole und von Codeaufrufen an die AWS IoT Events APIs.

Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen fürAWS IoT Events. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage ermitteln CloudTrail, an die die Anfrage gestellt wurdeAWS IoT Events, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrailBenutzerhandbuch.

AWS IoT EventsInformationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn eine Aktivität in stattfindetAWS IoT Events, wird diese Aktivität zusammen mit anderen CloudTrail AWS Serviceereignissen in der Ereignishistorie aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Zur kontinuierlichen Aufzeichnung von Ereignissen in Ihrem AWS-Konto, einschließlich Ereignissen für AWS IoT Events, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket bereit. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- CloudTrail unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Gibt an, ob die Anfrage mit Root- oder IAM-Benutzer-Anmeldeinformationen von ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen verbundenen Benutzer gesendet wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter dem CloudTrail UserIdentity-Element. AWS IoT EventsAktionen sind in der API-Referenz dokumentiert. AWS IoT Events

Grundlagen zu AWS IoT Events-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. AWS CloudTrailProtokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Wenn die CloudTrail Protokollierung in Ihrem AWS Konto aktiviert ist, werden die meisten API-Aufrufe von AWS IoT Events Aktionen in CloudTrail Protokolldateien aufgezeichnet, wo sie zusammen mit

anderen AWS Serviceaufzeichnungen geschrieben werden. CloudTrail bestimmt anhand eines Zeitraums und der Dateigröße, wann eine neue Datei erstellt und in sie geschrieben werden soll.

Jeder Protokolleintrag enthält Informationen über den Ersteller der Anforderung. Der Benutzeridentität im Protokolleintrag können Sie folgende Informationen entnehmen:

- Gibt an, ob die Anfrage mit Root- oder IAM-Benutzer-Anmeldeinformationen von ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen verbundenen Benutzer gesendet wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Sie können Ihre Protokolldateien so lange in Ihrem Amazon S3 S3-Bucket speichern, wie Sie möchten, aber Sie können auch Amazon S3 S3-Lebenszyklusregeln definieren, um Protokolldateien automatisch zu archivieren oder zu löschen. Standardmäßig werden Ihre Protokolldateien mit der serverseitigen Verschlüsselung (SSE) von Amazon S3 verschlüsselt.

Um bei der Übermittlung der Protokolldatei benachrichtigt zu werden, können Sie so konfigurieren, CloudTrail dass Amazon SNS SNS-Benachrichtigungen veröffentlicht werden, wenn neue Protokolldateien zugestellt werden. Weitere Informationen finden Sie unter Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail.

Sie können die AWS IoT Events-Protokolldateien aus mehreren AWS-Regionen und AWS-Konten auch in einem einzigen Amazon-S3-Bucket zusammenfassen.

Weitere Informationen finden Sie unter Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DescribeDetector` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/bertholt-brecht",
    "accountId": "123456789012",
    "accessKeyId": "access-key-id",
    "sessionContext": {
      "attributes": {
```

```
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-08T18:53:58Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2019-02-08T19:02:44Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeDetector",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-cli/1.15.65 Python/3.7.1 Darwin/16.7.0 botocore/1.10.65",
  "requestParameters": {
    "detectorModelName": "pressureThresholdEventDetector-brecht",
    "keyValue": "1"
  },
  "responseElements": null,
  "requestID": "00f41283-ea0f-4e85-959f-bee37454627a",
  "eventID": "5eb0180d-052b-49d9-a289-0eb8d08d4c27",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateDetectorModel` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-Lambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEvents-RoleForIotEvents-
ABC123DEF456/IotEvents-Lambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
```

```
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABC123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:43Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "CreateDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel",
    "key": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "roleArn": "arn:aws:iam::123456789012:role/events_action_execution_role"
  },
  "responseElements": null,
  "requestID": "cecfbfa1-e452-4fa6-b86b-89a89f392b66",
  "eventID": "8138d46b-50a3-4af0-9c5e-5af5ef75ea55",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateInput` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-Lambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456/IotEvents-Lambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
```

```
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABC123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:43Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "CreateInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "inputName": "batchputmessagedetectorupdated",
    "inputDescription": "batchputmessagedetectorupdated"
  },
  "responseElements": null,
  "requestID": "fb315af4-39e9-4114-94d1-89c9183394c1",
  "eventID": "6d8cf67b-2a03-46e6-bbff-e113a7bded1e",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DeleteDetectorModel` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
```

```
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:11Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DeleteDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel"
  },
  "responseElements": null,
  "requestID": "149064c1-4e24-4160-a5b2-1065e63ee2e4",
  "eventID": "7669db89-dcc0-4c42-904b-f24b764dd808",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DeleteInput` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
```

```
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:38Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DeleteInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "errorCode": "ResourceNotFoundException",
  "errorMessage": "Input of name: NoSuchInput not found",
  "requestParameters": {
    "inputName": "NoSuchInput"
  },
  "responseElements": null,
  "requestID": "ce6d28ac-5baf-423d-a5c3-afd009c967e3",
  "eventID": "be0ef01d-1c28-48cd-895e-c3ff3172c08e",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DescribeDetectorModel` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
```

```
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AAKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:20Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel"
  },
  "responseElements": null,
  "requestID": "18a11622-8193-49a9-85cb-1fa6d3929394",
  "eventID": "1ad80ff8-3e2b-4073-ac38-9cb3385beb04",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DescribeInput` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
```

```
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AAKIAI44QH8DHBEXAMPLE",
    "sessionContext": {

      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:56:09Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "inputName": "input_createinput"
  },
  "responseElements": null,
  "requestID": "3af641fa-d8af-41c9-ba77-ac9c6260f8b8",
  "eventID": "bc4e6cc0-55f7-45c1-b597-ec99aa14c81a",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `DescribeLoggingOptions` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
```

```
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:23Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeLoggingOptions",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "b624b6c5-aa33-41d8-867b-025ec747ee8f",
  "eventID": "9c7ce626-25c8-413a-96e7-92b823d6c850",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `ListDetectorModels` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
```

```
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:23Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectorModels",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "nextToken": "CkZEZXRlY3Rvck1vZGVsMl9saXN0ZGV0ZWN0b3Jtb2RlbHN0ZXN0X2VlOWJkZTk1YT",
    "maxResults": 3
  },
  "responseElements": null,
  "requestID": "6d70f262-da95-4bb5-94b4-c08369df75bb",
  "eventID": "2d01a25c-d5c7-4233-99fe-ce1b8ec05516",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `ListDetectorModelVersions` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
```

```
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:33Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectorModelVersions",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel",
    "maxResults": 2
  },
  "responseElements": null,
  "requestID": "ebecb277-6bd8-44ea-8abd-fbf40ac044ee",
  "eventID": "fc6281a2-3fac-4e1e-98e0-ca6560b8b8be",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `ListDetectors` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
```

```
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:54Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectors",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "batchputmessagedetectorinstancecreated",
    "stateName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "responseElements": null,
  "requestID": "4783666d-1e87-42a8-85f7-22d43068af94",
  "eventID": "0d2b7e9b-afe6-4aef-afd2-a0bb1e9614a9",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `ListInputs` Aktion demonstriert.

```
{
  "eventVersion": "1.05",
```

```
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:57Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListInputs",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "nextToken": "CkhjYW5hcnlfdGVzdF9pbnB1dF9saXN0ZGV0ZWN0b3Jtb2RlbHN0ZXN0ZDU3OGZ",
    "maxResults": 3
  },
  "responseElements": null,
  "requestID": "dd6762a1-1f24-4e63-a986-5ea3938a03da",
  "eventID": "c500f6d8-e271-4366-8f20-da4413752469",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `PutLoggingOptions` Aktion demonstriert.

```
{
```

```
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:56:43Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "PutLoggingOptions",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "loggingOptions": {
      "roleArn": "arn:aws:iam::123456789012:role/logging__logging_role",
      "level": "INFO",
      "enabled": false
    }
  },
  "responseElements": null,
  "requestID": "df570e50-fb19-4636-9ec0-e150a94bc52c",
  "eventID": "3247f928-26aa-471e-b669-e4a9e6fbc42c",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `UpdateDetectorModel` Aktion demonstriert.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:55:51Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "UpdateDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel",
    "roleArn": "arn:aws:iam::123456789012:role/Events_action_execution_role"
  },
  "responseElements": null,
  "requestID": "add29860-c1c5-4091-9917-d2ef13c356cf",
  "eventID": "7baa9a14-6a52-47dc-aea0-3cace05147c3",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `UpdateInput` Aktion demonstriert.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:00Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "UpdateInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "errorCode": "ResourceNotFoundException",
  "errorMessage": "Input of name: NoSuchInput not found",
  "requestParameters": {
    "inputName": "NoSuchInput",
    "inputDescription": "this is a description of an input"
  },
  "responseElements": null,
  "requestID": "58d5d2bb-4110-4c56-896a-ee9156009f41",
  "eventID": "c2df241a-fd53-4fd0-936c-ba309e5dc62d",
  "eventType": "AwsApiCall",
```

```
  "recipientAccountId": "123456789012"
}
```

Compliance-Validierung für AWS IoT Events

Informationen darüber, ob ein AWS-Service in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter AWS-Services in Geltungsbereich nach Compliance-Programm. Wählen Sie das Compliance-Programm, das Sie interessiert. Allgemeine Informationen finden Sie unter AWS-Compliance-Programme.

Sie können Auditberichte von Drittanbietern unter AWS Artifact herunterladen. Weitere Informationen finden Sie unter Berichte herunterladen in AWS Artifact.

Ihre Compliance-Verantwortung bei der Verwendung von AWS-Services ist von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften abhängig. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Kurzanleitungen für Sicherheit und Compliance: In diesen Bereitstellungsleitfäden werden Überlegungen zur Architektur erörtert und Schritte zum Bereitstellen von Basisumgebungen auf AWS zur Verfügung gestellt, die auf Sicherheit und Compliance ausgerichtet sind.
- Erstellung einer Architektur mit HIPAA-konformer Sicherheit und Compliance in Amazon Web Services – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe von AWS HIPAA-berechtigte Anwendungen erstellen können.

  > **Note**
  >
  > Nicht alle AWS-Services sind HIPAA-berechtigt. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS-Compliance-Ressourcen – Diese Arbeitsbücher und Leitfäden könnten für Ihre Branche und Ihren Standort relevant sein.
- AWS-Compliance-Leitfäden für Kunden: Verstehen Sie das Modell der geteilten Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Methoden zum Schutz von AWS-Services zusammengefasst und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.

- Auswertung von Ressourcen mit Regeln im AWS ConfigEntwicklerhandbuch – Der AWS Config-Service bewertet, wie gut Ihre Ressourcenkonfigurationen mit internen Praktiken, Branchenrichtlinien und Vorschriften übereinstimmen.
- AWS Security Hub: Dieser AWS-Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus innerhalb von AWS. Security Hub verwendet Sicherheitskontrollen, um Ihre AWS-Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- AWS Audit Manager: Dieser AWS-Service hilft Ihnen, Ihre AWS-Nutzung kontinuierlich zu überprüfen, um den Umgang mit Risiken und die Compliance von Branchenstandards zu vereinfachen.

Ausfallsicherheit in AWS IoT Events

Im Zentrum der globalen AWS-Infrastruktur stehen die AWS-Regionen und Availability Zones. AWS Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die mit Netzwerken mit geringer Latenz, hohem Durchsatz und hochredundanten Vernetzungen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen über AWS-Regionen und -Availability Zones finden Sie unter Globale AWS-Infrastruktur.

Sicherheit der Infrastruktur in AWS IoT Events

Als verwalteter Service ist AWS IoT Events durch die globalen Verfahren zur Gewährleistung der Netzwerksicherheit von AWS geschützt. Informationen zu AWS-Sicherheitsdiensten und wie AWS die Infrastruktur schützt, finden Sie unter AWSCloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS-Umgebung anhand der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz im Security Pillar AWS Well-Architected Framework.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

AWS IoT Events-Kontingente

Das Allgemeine AWS-ReferenzHandbuch enthält die Standardkontingente AWS IoT Events für ein AWS Konto. Sofern nicht anders angegeben, gilt jedes Kontingent pro AWS Region. Weitere Informationen finden Sie im Allgemeine AWS-ReferenzHandbuch unter AWS IoT EventsEndpunkte und Kontingente und AWSService Quotas.

Um eine Erhöhung des Servicekontingents zu beantragen, reichen Sie in der Support Center-Konsole eine Support-Anfrage ein. Weitere Informationen finden Sie unter Beantragen einer Quota-Erhöhung im Service-Quotas-Benutzerhandbuch.

> **Note**
>
> - Alle Namen für Meldermodelle und Eingänge müssen innerhalb eines Kontos eindeutig sein.
> - Sie können die Namen von Meldermodellen und Eingängen nicht mehr ändern, nachdem sie erstellt wurden.

Markieren Ihrer AWS IoT Events-Ressourcen

Um Ihnen bei der Verwaltung und Organisation Ihrer Detektormodelle und Eingänge zu helfen, können Sie optional jeder dieser Ressourcen Ihre eigenen Metadaten in Form von Tags zuweisen. In diesem Abschnitt werden Tags und deren Erstellung beschrieben.

Grundlagen zu Tags (Markierungen)

Mit Tags (Markierungen) können Sie Ihre AWS IoT Events-Ressourcen auf unterschiedliche Weise kategorisieren (z. B. nach Zweck, Eigentümer oder Umgebung). Dies ist nützlich, wenn Sie viele Ressourcen desselben Typs haben. Sie können eine bestimmte Ressource anhand der ihr zugewiesenen Tags schnell identifizieren.

Jedes Tag besteht aus einem Schlüssel und einem optionalen Wert, die Sie beide selbst definieren können. Sie könnten beispielsweise eine Reihe von Tags für Ihre Eingaben definieren, mit deren Hilfe Sie die Geräte, die diese Eingaben senden, anhand ihres Typs verfolgen können. Wir empfehlen die Erstellung von Tag-Schlüsseln, die die Anforderungen der jeweiligen Ressourcenart erfüllen. Eine Anzahl einheitlicher Tag (Markierung)-Schlüssel vereinfacht das Verwalten der Ressourcen.

Sie können anhand der von Ihnen hinzugefügten oder angewendeten Tags nach Ressourcen suchen und diese filtern, Tags verwenden, um Ihre Kosten zu kategorisieren und nachzuverfolgen, und auch Tags verwenden, um den Zugriff auf Ihre Ressourcen zu kontrollieren, wie unter Verwenden von Tags mit IAM-Richtlinien im AWS IoTEntwicklerhandbuch beschrieben.

Um die Bedienung zu vereinfachen, AWS Management Console bietet der Tag-Editor im eine zentrale, einheitliche Möglichkeit, Ihre Tags zu erstellen und zu verwalten. Weitere Informationen finden Sie unter Arbeiten mit dem Tag-Editor in Arbeiten mit dem AWS Management Console.

Sie können auch mit Tags arbeiten, indem Sie die AWS CLI und die AWS IoT Events API verwenden. Sie können Tags bei der Erstellung mit Detektormodellen und Eingaben verknüpfen, indem Sie das `"Tags"` Feld in den folgenden Befehlen verwenden:

- CreateDetectorModel
- CreateInput

Sie können Tags für vorhandene Ressourcen, die das Markieren unterstützen, hinzufügen, ändern oder löschen. Verwenden Sie dazu die folgenden Befehle:

- TagResource
- ListTagsForResource
- UntagResource

Sie können Tag (Markierung)-Schlüssel und -Werte bearbeiten und Tags (Markierungen) jederzeit von einer Ressource entfernen. Sie können den Wert eines Tags (Markierung) zwar auf eine leere Zeichenfolge, jedoch nicht Null festlegen. Wenn Sie ein Tag (Markierung) mit demselben Schlüssel wie ein vorhandener Tag (Markierung) für die Ressource hinzufügen, wird der alte Wert mit dem neuen überschrieben. Wenn Sie eine Ressource löschen, werden alle der Ressource zugeordneten Tags ebenfalls gelöscht.

Zusätzliche Informationen finden Sie unter AWSTagging-Strategien.

Tag-Beschränkungen und -Einschränkungen

Die folgenden grundlegenden Einschränkungen gelten für Tags (Markierungen):

- Maximale Anzahl von Tags (Markierungen) pro Ressource: 50
- Maximale Schlüssellänge — 127 Unicode-Zeichen in UTF-8
- Maximale Wertlänge — 255 Unicode-Zeichen in UTF-8
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Verwenden Sie das `"aws:"` Präfix nicht in Ihren Tagnamen oder -Werten, da es für AWS die Verwendung reserviert ist. Sie können keine Tag-Namen oder Werte mit diesem Präfix bearbeiten oder löschen. Tags mit diesem Präfix werden nicht zum Limit für Tags pro Ressource gezählt.
- Wenn Ihr Markierungsschema für mehrere -Services und -Ressourcen verwendet wird, denken Sie daran, dass andere Services möglicherweise Einschränkungen für zulässige Zeichen haben. Im allgemeinen zulässige Zeichen: Buchstaben, Leerzeichen und Zahlen, die in UTF-8 darstellbar sind, sowie die folgenden Sonderzeichen: + - = . _ : / @.

Verwenden von Tags mit IAM-Richtlinien

Sie können Tag-basierte Berechtigungen auf Ressourcenebene in den IAM-Richtlinien anwenden, die Sie für AWS IoT Events-API-Aktionen verwenden. Dies ermöglicht Ihnen eine bessere Kontrolle darüber, welche Ressourcen ein Benutzer erstellen, ändern oder verwenden kann.

Sie können das `Condition`-Element (auch als `Condition`-Block bezeichnet) mit den folgenden Bedingungskontextschlüsseln und Werten in einer IAM-Richtlinie zum Steuern des Benutzerzugriffs (Berechtigungen) basierend auf den Tags einer Ressource verwenden:

- Verwenden Sie `aws:ResourceTag/<tag-key>: <tag-value>`, um Benutzeraktionen für Ressourcen mit bestimmten Tags zuzulassen oder zu verweigern.
- Verwenden Sie `aws:RequestTag/<tag-key>: <tag-value>`, um festzulegen, dass ein bestimmtes Tag verwendet (oder nicht verwendet) wird, wenn Sie eine API-Anfrage stellen, um eine Ressource zu erstellen oder zu ändern, die Tags zulässt.
- Verwenden Sie `aws:TagKeys: [<tag-key>, ...]`, um zu verlangen, dass ein bestimmter Satz von Tag-Schlüsseln verwendet wird (oder nicht), wenn eine API-Anforderung zum Erstellen einer Ressource durchgeführt wird, die Tags zulässt.

> **Note**
>
> Die Bedingungskontextschlüssel und -werte in einer IAM-Richtlinie gelten nur für die AWS IoT Events-Aktionen, bei denen eine Kennung für eine Ressource, die Tags zulässt, ein erforderlicher Parameter ist.

Im AWS Identity and Access ManagementBenutzerhandbuch finden Sie zusätzliche Informationen zur Verwendung von Tags zur Zugriffssteuerung mithilfe von Tags. Der Referenzabschnitt zu den IAM-JSON-Richtlinien dieses Handbuchs enthält ausführliche Syntax, Beschreibungen und Beispiele der Elemente, Variablen und Bewertungslogik von JSON-Richtlinien in IAM.

Die folgende Beispielrichtlinie wendet zwei auf Tags basierende Einschränkungen an. Ein Benutzer, der durch diese Richtlinie eingeschränkt ist:

- Kann keiner Ressource den Tag „env = prod“ zuweisen (im Beispiel vgl. die Zeile `"aws:RequestTag/env" : "prod"`
- Kann keine Ressource modifizieren oder darauf zugreifen, die den Tag „env=prod“ aufweist (im Beispiel vgl. die Zeile `"aws:ResourceTag/env" : "prod"`).

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

```
            "Effect": "Deny",
            "Action": [
                "iotevents:CreateDetectorModel",
                "iotevents:CreateAlarmModel",
                "iotevents:CreateInput",
                "iotevents:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/env": "prod"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iotevents:DescribeDetectorModel",
                "iotevents:DescribeAlarmModel",
                "iotevents:UpdateDetectorModel",
                "iotevents:UpdateAlarmModel",
                "iotevents:DeleteDetectorModel",
                "iotevents:DeleteAlarmModel",
                "iotevents:ListDetectorModelVersions",
                "iotevents:ListAlarmModelVersions",
                "iotevents:UpdateInput",
                "iotevents:DescribeInput",
                "iotevents:DeleteInput",
                "iotevents:ListTagsForResource",
                "iotevents:TagResource",
                "iotevents:UntagResource",
                "iotevents:UpdateInputRouting"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/env": "prod"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotevents:*"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Sie können auch mehrere Tag-Werte für einen bestimmten Tag-Schlüssel angeben, indem Sie sie wie folgt in eine Liste einschließen.

```
            "StringEquals" : {
              "aws:ResourceTag/env" : ["dev", "test"]
            }
```

> **Note**
>
> Wenn Sie Benutzern den Zugriff zu Ressourcen auf der Grundlage von Tags (Markierungen) gewähren oder verweigern, müssen Sie daran denken, Benutzern explizit das Hinzufügen und Entfernen dieser Tags (Markierungen) von den jeweiligen Ressourcen unmöglich zu machen. Andernfalls können Benutzer möglicherweise Ihre Einschränkungen umgehen und sich Zugriff auf eine Ressource verschaffen, indem sie ihre Tags (Markierungen) modifizieren.

Fehlerbehebung für AWS IoT Events

Verwenden Sie die Informationen in diesen Abschnitten, um Probleme mit AWS IoT Events zu beheben und zu lösen.

Themen

Häufige AWS IoT Events Probleme und Lösungen

Im folgenden Abschnitt finden Sie Informationen zur Fehlerbehebung und finden mögliche Lösungen zur Behebung von Problemen mit AWS IoT Events.

Fehler

Fehler bei der Erstellung des Detector-Modells

Ich erhalte Fehler, wenn ich versuche, ein Detektormodell zu erstellen.

Lösung

Wenn Sie ein Detektormodell erstellen, müssen Sie die folgenden Einschränkungen berücksichtigen.

- In jedem `action` Feld ist nur eine Aktion zulässig.
- Die `condition` ist für erforderlich`transitionEvents`. Es ist optional für `OnEnter`-`OnInput`, - und -`OnExit`Ereignisse.
- Wenn das `condition` Feld leer ist, entspricht das ausgewertete Ergebnis des Bedingungsausdrucks `true`.
- Das ausgewertete Ergebnis des Bedingungsausdrucks sollte ein boolescher Wert sein. Wenn das Ergebnis kein boolescher Wert ist, entspricht er `false` und löst den - `actions` oder -Übergang zu dem im Ereignis `nextState` angegebenen nicht aus.

Weitere Informationen finden Sie unter Einschränkungen und Einschränkungen des Detektormodells.

Aktualisierungen aus einem gelöschten Detektormodell

Ich habe vor einigen Minuten ein Detektormodell aktualisiert oder gelöscht, erhalte aber immer noch Statusaktualisierungen vom alten Detektormodell über MQTT-Nachrichten oder SNS-Warnungen.

Lösung

Wenn Sie ein Detektormodell aktualisieren, löschen oder neu erstellen (siehe UpdateDetectorModel), gibt es eine Verzögerung, bevor alle Detektor-Instances gelöscht und das neue Modell verwendet wird. Während dieser Zeit werden Eingaben möglicherweise weiterhin von den Instances der vorherigen Version des Detektormodells verarbeitet. Möglicherweise erhalten Sie weiterhin Warnungen, die vom vorherigen Detektormodell definiert wurden. Warten Sie mindestens sieben Minuten, bevor Sie die Aktualisierung erneut überprüfen oder einen Fehler melden.

Fehler beim Aktionsauslöser (bei Erfüllen einer Bedingung)

Der Detektor kann keine Aktion auslösen oder in einen neuen Zustand übergehen, wenn die Bedingung erfüllt ist.

Lösung

Stellen Sie sicher, dass das ausgewertete Ergebnis des bedingten Ausdrucks des Detektors ein boolescher Wert ist. Wenn es sich bei dem Ergebnis nicht um einen booleschen Wert handelt, entspricht er dem im Ereignis `nextState` angegebenen `false` Wert und löst den Übergang `action` oder nicht aus. Weitere Informationen finden Sie unter Syntax für bedingte Ausdrücke.

Fehler beim Aktionsauslöser (beim Überschreiten eines Schwellenwerts)

Der Detektor löst keine Aktion oder einen Ereignisübergang aus, wenn die Variable in einem bedingten Ausdruck einen bestimmten Wert erreicht.

Lösung

Wenn Sie `setVariable` für `onInput`, `onEnter`oder aktualisieren`onExit`, wird der neue Wert während `condition` des aktuellen Verarbeitungszyklus nicht verwendet. Stattdessen wird der ursprüngliche Wert verwendet, bis der aktuelle Zyklus abgeschlossen ist. Sie können dieses Verhalten ändern, indem Sie den `evaluationMethod` Parameter in der Detektormodelldefinition festlegen. Wenn auf festgelegt `evaluationMethod` ist`SERIAL`, werden Variablen aktualisiert und Ereignisbedingungen in der Reihenfolge ausgewertet, in der die Ereignisse definiert sind. Wenn auf `BATCH` (Standard) gesetzt `evaluationMethod` ist, werden Variablen aktualisiert und Ereignisse werden erst ausgeführt, nachdem alle Ereignisbedingungen ausgewertet wurden.

Falsche Statusnutzung

Der Detektor wechselt in den falschen Zustand, wenn ich versuche, Nachrichten mithilfe von an Eingaben zu sende`BatchPutMessage`.

Lösung

Wenn Sie verwenden, BatchPutMessage um mehrere Nachrichten an Eingaben zu senden, ist die Reihenfolge, in der die Nachrichten oder Eingaben verarbeitet werden, nicht garantiert. Um die Reihenfolge zu gewährleisten, senden Sie Nachrichten einzeln und warten Sie jedes Mal, `BatchPutMessage` bis den Erfolg bestätigt hat.

Verbindungsnachricht

Ich erhalte eine `('Connection aborted.', error(54, 'Connection reset by peer'))` Fehlermeldung, wenn ich versuche, eine API aufzurufen oder aufzurufen.

Lösung

Stellen Sie sicher, dass OpenSSL TLS 1.1 oder eine neuere Version verwendet, um die Verbindung herzustellen. Dies sollte bei den meisten Linux-Distributionen oder Windows-Version 7 und höher der Standardwert sein. Benutzer von macOS müssen möglicherweise OpenSSL aktualisieren.

InvalidRequestException Nachricht

Ich erhalte InvalidRequestException , wenn ich versuche, die - `CreateDetectorModel` und -`UpdateDetectorModel`APIs aufzurufen. APIs

Lösung

Überprüfen Sie Folgendes, um das Problem zu beheben. Weitere Informationen erhalten Sie unter CreateDetectorModel und UpdateDetectorModel.

- Stellen Sie sicher, dass Sie nicht sowohl `seconds` als auch `durationExpression` als Parameter von `SetTimerAction` gleichzeitig verwenden.
- Stellen Sie sicher, dass Ihr Zeichenfolgenausdruck für gültig `durationExpression` ist. Der Zeichenfolgenausdruck kann Zahlen, Variablen (`$variable.<variable-name>`) oder Eingabewerte () enthalten`$input.<input-name>.<path-to-datum>`.

Amazon- CloudWatch Logs-**`action.setTimer`**Fehler

Sie können Amazon CloudWatch Logs einrichten, um AWS IoT EventsDetektormodell-Instances zu überwachen. Im Folgenden finden Sie häufige Fehler, die von generiert werdenAWS IoT Events, wenn Sie verwendenaction.setTimer.

- Fehler: Ihr Dauerausdruck für den Timer mit dem Namen *`<timer-name>`* konnte nicht mit einer Zahl ausgewertet werden.

Lösung

Stellen Sie sicher, dass Ihr Zeichenfolgenausdruck für in eine Zahl konvertiert werden `durationExpression` kann. Andere Datentypen, z. B. boolean, sind nicht zulässig.

- Fehler: Das ausgewertete Ergebnis Ihres Dauerausdrucks für den Timer mit dem Namen *`<timer-name>`* ist größer als 31622440. Um die Genauigkeit zu gewährleisten, stellen Sie sicher, dass sich Ihr Dauerausdruck auf einen Wert zwischen 60 und 31622400 bezieht.

Lösung

Stellen Sie sicher, dass die Dauer Ihres Timers höchstens 31622400 Sekunden beträgt. Das ausgewertete Ergebnis der Dauer wird auf die nächste ganze Zahl abgerundet.

- Fehler: Das ausgewertete Ergebnis Ihres Dauerausdrucks für den Timer mit dem Namen *<timer-name>* ist kleiner als 60. Um die Genauigkeit zu gewährleisten, stellen Sie sicher, dass sich Ihr Dauerausdruck auf einen Wert zwischen 60 und 31622400 bezieht.

Lösung

Stellen Sie sicher, dass die Dauer Ihres Timers mindestens 60 Sekunden beträgt. Das ausgewertete Ergebnis der Dauer wird auf die nächste ganze Zahl abgerundet.

- Fehler: Ihr Dauerausdruck für den Timer mit dem Namen *<timer-name>* konnte nicht ausgewertet werden. Überprüfen Sie die Variablennamen, Eingabenamen und Pfade zu den Daten, um sicherzustellen, dass Sie auf die vorhandenen Variablen und Eingaben verweisen.

Lösung

Stellen Sie sicher, dass sich Ihr Zeichenfolgenausdruck auf die vorhandenen Variablen und Eingaben bezieht. Der Zeichenfolgenausdruck kann Zahlen, Variablen (`$variable.`*variable-name*) und Eingabewerte () enthalten`$input.`*input-name*`.`*path-to-datum*.

- Fehler: Der Timer mit dem Namen konnte nicht festgelegt werden`<timer-name>`. Überprüfen Sie Ihren Dauerausdruck und versuchen Sie es erneut.

Lösung

Sehen Sie sich die SetTimerAction Aktion an, um sicherzustellen, dass Sie die richtigen Parameter angegeben haben, und legen Sie dann den Timer erneut fest.

Weitere Informationen finden Sie unter Aktivieren der Amazon- CloudWatch Protokollierung bei der Entwicklung von AWS IoT EventsDetektormodellen.

Amazon CloudWatch -Nutzlastfehler

Sie können Amazon CloudWatch Logs einrichten, um AWS IoT EventsDetektormodell-Instances zu überwachen. Im Folgenden finden Sie häufige Fehler und Warnungen, die von generiert werdenAWS IoT Events, wenn Sie die Aktionsnutzlast konfigurieren.

- Fehler: Wir konnten Ihren Ausdruck für die Aktion nicht auswerten. Stellen Sie sicher, dass sich die Variablennamen, Eingabenamen und Pfade zu den Daten auf die vorhandenen Variablen und Eingabewerte beziehen. Überprüfen Sie außerdem, ob die Größe der Nutzlast weniger als 1 KB beträgt, die maximal zulässige Größe einer Nutzlast.

Lösung

Stellen Sie sicher, dass Sie die richtigen Variablennamen, Eingabenamen und Pfade zu den Daten eingeben. Sie erhalten diese Fehlermeldung möglicherweise auch, wenn die Aktionsnutzlast größer als 1 KB ist.

- Fehler: Wir konnten Ihren Inhaltsausdruck für die Nutzlast von nicht analysieren*<action-type>*. Geben Sie einen Inhaltsausdruck mit der richtigen Syntax ein.

Lösung

Der Inhaltsausdruck kann Zeichenfolgen (`'`*`string`*`'`), Variablen (`$variable.`*`variable-name`*), Eingabewerte (`$input.`*`input-name`*`.`*`path-to-datum`*), Zeichenfolgenverkettungen und Zeichenfolgen enthalten, die enthalten`${}`.

- Fehler: Ihr Nutzlastausdruck {*`expression`* } ist ungültig. Der definierte Nutzlasttyp ist JSON, daher müssen Sie einen Ausdruck angeben, der zu einer Zeichenfolge ausgewertet AWS IoT Events wird.

Lösung

Wenn der angegebene Nutzlasttyp JSON ist, prüft AWS IoT Events zunächst, ob der Service Ihren Ausdruck in eine Zeichenfolge auswerten kann. Das ausgewertete Ergebnis darf keine boolesche Zahl oder Zahl sein. Wenn die Validierung fehlschlägt, erhalten Sie diesen Fehler möglicherweise.

- Warnung: Die Aktion wurde ausgeführt, aber wir konnten Ihren Inhaltsausdruck für die Aktionsnutzlast nicht in gültiges JSON auswerten. Der definierte Nutzlasttyp ist JSON.

Lösung

Stellen Sie sicher, dass Ihren Inhaltsausdruck für die Aktionsnutzlast auf gültiges JSON auswerten AWS IoT Events kann, wenn Sie den Nutzlasttyp als definieren`JSON`. AWS IoT Events führt die Aktion aus, auch wenn den Inhaltsausdruck nicht auf gültiges JSON auswerten AWS IoT Events kann.

Weitere Informationen finden Sie unter Aktivieren der Amazon- CloudWatch Protokollierung bei der Entwicklung von AWS IoT EventsDetektormodellen.

Inkompatible Datentypen

Nachricht: Inkompatible Datentypen [`<inferred-types>`], die für `<reference>` im folgenden Ausdruck gefunden wurden: `<expression>`

Lösung

Dieser Fehler kann aus einem der folgenden Gründe auftreten:

- Die ausgewerteten Ergebnisse Ihrer Referenzen sind nicht mit anderen Operanden in Ihren Ausdrücken kompatibel.
- Der Typ des Arguments, das an eine Funktion übergeben wird, wird nicht unterstützt.

Wenn Sie Verweise in Ausdrücken verwenden, überprüfen Sie Folgendes:

- Wenn Sie eine Referenz als Operanden mit einem oder mehreren Operatoren verwenden, stellen Sie sicher, dass alle Datentypen, auf die Sie verweisen, kompatibel sind.

  Im folgenden Ausdruck 2 ist Ganzzahl beispielsweise ein Operand der && Operatoren == und . Um sicherzustellen, dass die Operanden kompatibel sind, `$variable.testVariable` müssen `$variable.testVariable + 1` sie auf eine Ganzzahl oder Dezimalzahl verweisen.

  Darüber hinaus 1 ist Ganzzahl ein Operand des -+Operators. Daher `$variable.testVariable` muss auf eine Ganzzahl oder Dezimalzahl verweisen.

  ```
  ‘$variable.testVariable + 1 == 2 && $variable.testVariable’
  ```

- Wenn Sie eine Referenz als Argument verwenden, das an eine Funktion übergeben wird, stellen Sie sicher, dass die Funktion die Datentypen unterstützt, auf die Sie verweisen.

  Die folgende `timeout("time-name")` Funktion erfordert beispielsweise eine Zeichenfolge mit doppelten Anführungszeichen als Argument. Wenn Sie eine Referenz für den Wert *`timer-name`* verwenden, müssen Sie eine Zeichenfolge mit doppelten Anführungszeichen referenzieren.

  ```
  timeout("timer-name")
  ```

> **Note**
>
> Wenn Sie für die `convert(type, expression)` Funktion eine Referenz für den *`Typwert`* verwenden, muss das ausgewertete Ergebnis Ihrer Referenz `StringDecimal`, oder sein`Boolean`.

Weitere Informationen finden Sie unter Referenzen.

Nachricht konnte nicht an gesendet werden AWS IoT Events

Nachricht: Nachricht konnte nicht an Iot Events gesendet werden

Lösung

Dieser Fehler kann aus folgenden Gründen auftreten:

- Die Nutzlast der Eingabenachricht enthält nicht `Input attribute Key`.
- Die `Input attribute Key` befindet sich nicht im selben JSON-Pfad wie in der Eingabedefinition angegeben.
- Die Eingabenachricht stimmt nicht mit dem Schema überein, wie in der AWS IoT Events Eingabe definiert.

> **Note**
>
> Bei der Datenaufnahme von anderen -Services kommt es ebenfalls zu einem Fehler.

Example

In schlägt AWS IoT Coredie AWS IoT Regel beispielsweise mit der folgenden Meldung fehl `Verify the Input Attribute key.`

Um dies zu beheben, stellen Sie sicher, dass das Eingabe-Nutzlast-Nachrichtenschema der AWS IoT Events Eingabedefinition und der `Input attribute Key` Speicherort übereinstimmt. Weitere Informationen finden Sie unter , um the section called “Eine Eingabe im Navigationsbereich erstellen” zu erfahren, wie Sie AWS IoT Events Eingaben definieren.

Fehlerbehebung bei einem Detektormodell durch Ausführen von Analysen

AWS IoT Eventskann Ihr Detektormodell analysieren und Analyseergebnisse generieren, ohne Eingabedaten an Ihr Detektormodell zu senden. AWS IoT Eventsführt eine Reihe von Analysen durch, die in diesem Abschnitt beschrieben werden, um Ihr Detektormodell zu überprüfen. Diese erweiterte Lösung zur Fehlerbehebung fasst auch Diagnoseinformationen zusammen, einschließlich Schweregrad und Lokalisation, sodass Sie potenzielle Probleme in Ihrem Meldermodell schnell finden und beheben können. Weitere Informationen zu Diagnosefehlertypen und Meldungen für Ihr Meldermodell finden Sie unterAnalyse- und Diagnoseinformationen für Detektormodelle.

Sie können die AWS IoT Events Konsole, API, AWS Command Line Interface(AWS CLI) oder AWSdas SDK verwenden, um diagnostische Fehlermeldungen aus der Analyse Ihres Meldermodells anzuzeigen.

> **Note**
>
> - Sie müssen alle Fehler beheben, bevor Sie Ihr Meldermodell veröffentlichen können.
> - Wir empfehlen Ihnen, die Warnungen zu lesen und die erforderlichen Maßnahmen zu ergreifen, bevor Sie Ihr Detektormodell in Produktionsumgebungen verwenden. Andernfalls funktioniert das Meldermodell möglicherweise nicht wie erwartet.
> - Sie können bis zu 10 Analysen gleichzeitig im `RUNNING` Status haben.

Informationen zur Analyse Ihres Detektormodells finden Sie unter Analysieren eines Detektormodells (Konsole) oderAnalysieren eines Detektormodells (AWS CLI).

Themen

Analyse- und Diagnoseinformationen für Detektormodelle

Detektormodellanalysen sammeln die folgenden Diagnoseinformationen:

- Stufe – Der Schweregrad des Analyseergebnisses. Je nach Schweregrad lassen sich die Analyseergebnisse in drei allgemeine Kategorien einteilen:
  - Information en(`INFO`) – Ein Informationsergebnis informiert Sie über ein wichtiges Feld in Ihrem Detektormodell. Diese Art von Ergebnis erfordert in der Regel keine sofortige Aktion.
  - Warnung (`WARNING`) – Ein Warnergebnis stuft Felder ein, die zu Problemen für Ihr Detektormodell führen können. Wir empfehlen Ihnen, Warnungen zu überprüfen und die erforderlichen Maßnahmen zu ergreifen, bevor Sie Ihr Detektormodell in Produktionsumgebungen verwenden. Andernfalls funktioniert das Detektormodell möglicherweise nicht wie erwartet.
  - Fehler (`ERROR`) – Ein Fehlerergebnis benachrichtigt Sie über ein Problem in Ihrem Detektormodell. führt AWS IoT Events automatisch diese Analysen durch, wenn Sie versuchen, das Detektormodell zu veröffentlichen. Sie müssen alle Fehler beheben, bevor Sie das Detektormodell veröffentlichen können.
- Standort – Enthält Informationen, mit denen Sie das Feld in Ihrem Detektormodell finden können, auf das das Analyseergebnis verweist. Ein Speicherort umfasst in der Regel den Statusnamen, den Namen des Übergangsereignisses, den Ereignisnamen und den Ausdruck (z. B. `in state TemperatureCheck in onEnter in event Init in action setVariable`).
- Typ – Der Typ des Analyseergebnisses. Analysetypen lassen sich in die folgenden Kategorien einteilen:
  - `supported-actions` – AWS IoT Events kann Aktionen aufrufen, wenn ein bestimmtes Ereignis oder Übergangsereignis erkannt wird. Sie können integrierte Aktionen definieren, um einen Timer zu verwenden, eine Variable festzulegen oder Daten an andere -AWSServices zu senden. Sie müssen Aktionen angeben, die mit anderen -AWSServices in einer -AWSRegion funktionieren, in der die AWS Services verfügbar sind.
  - `service-limits` – Service Quotas, auch als Limits bezeichnet, sind die maximale oder minimale Anzahl von Serviceressourcen oder -vorgängen für Ihr AWS Konto. Wenn nicht anders angegeben, gilt jedes Kontingent spezifisch für eine Region. Abhängig von Ihren Geschäftsanforderungen können Sie Ihr Detektormodell aktualisieren, um Limits zu vermeiden oder eine Kontingenterhöhung anzufordern. Sie können Erhöhungen für einige Kontingente beantragen, während andere Kontingente nicht erhöht werden können. Weitere Informationen finden Sie unter Kontingente.
- **`structure`** – Das Detektormodell muss über alle erforderlichen Komponenten wie Zustände verfügen und einer von AWS IoT Events unterstützten Struktur folgen. Ein Detektormodell muss mindestens einen Zustand und eine Bedingung haben, die die eingehenden Eingabedaten auswertet, um signifikante Ereignisse zu erkennen. Wenn ein Ereignis erkannt wird, wechselt das

Detektormodell in den nächsten Zustand und kann Aktionen aufrufen. Diese Ereignisse werden als Übergangsereignisse bezeichnet. Ein Übergangsereignis muss den nächsten Zustand anweisen, in den aufgenommen werden soll.

- **`expression-syntax`** – AWS IoT Events bietet mehrere Möglichkeiten, Werte anzugeben, wenn Sie Detektormodelle erstellen und aktualisieren. Sie können Literale, Operatoren, Funktionen, Referenzen und Ersatzvorlagen in den Ausdrücken verwenden. Sie können Ausdrücke verwenden, um Literalwerte anzugeben, oder die Ausdrücke AWS IoT Events auswerten, bevor Sie bestimmte Werte angeben. Ihr Ausdruck muss der erforderlichen Syntax entsprechen. Weitere Informationen finden Sie unter Ausdrücke.

  Detector Model-Ausdrücke in AWS IoT Events können auf bestimmte Daten oder eine Ressource verweisen.

  - **`data-type`** – AWS IoT Events unterstützt Ganzzahl-, Dezimal-, Zeichenfolgen- und boolesche Datentypen. Wenn die Daten eines Datentyps während der Ausdrucksauswertung automatisch in einen anderen konvertieren AWS IoT Events kann, sind diese Datentypen kompatibel.

    > **Note**
    >
    > - Ganzzahl und Dezimalzahl sind die einzigen kompatiblen Datentypen, die von unterstützt werdenAWS IoT Events.
    > - AWS IoT Events kann keine arithmetischen Ausdrücke auswerten, da eine Ganzzahl nicht in eine Zeichenfolge konvertieren AWS IoT Events kann.

  - **`referenced-data`** – Sie müssen die Daten definieren, auf die in Ihrem Detektormodell verwiesen wird, bevor Sie die Daten verwenden können. Wenn Sie beispielsweise Daten an eine DynamoDB-Tabelle senden möchten, müssen Sie eine Variable definieren, die auf den Tabellennamen verweist, bevor Sie die Variable in einem Ausdruck () verwenden können`$variable.TableName`.
  - **`referenced-resource`** – Ressourcen, die das Detektormodell verwendet, müssen verfügbar sein. Sie müssen Ressourcen definieren, bevor Sie sie verwenden können. Sie möchten beispielsweise ein Detektormodell erstellen, um die Temperatur eines Gewächshauses zu überwachen. Sie müssen eine Eingabe (`$input.TemperatureInput`) definieren, um eingehende Temperaturdaten an Ihr Detektormodell weiterzuleiten, bevor Sie die verwenden können`$input.TemperatureInput.sensorData.temperature`, um auf die Temperatur zu verweisen.

Im folgenden Abschnitt finden Sie Informationen zur Fehlerbehebung und mögliche Lösungen aus der Analyse Ihres Detektormodells.

Fehlerbehebung bei Fehlern beim Detektormodell

Die oben beschriebenen Fehlertypen liefern Diagnoseinformationen zu einem Detektormodell und entsprechen Nachrichten, die Sie möglicherweise abrufen. Verwenden Sie diese Meldungen und vorgeschlagenen Lösungen zur Behebung von Fehlern mit Ihrem Detektormodell.

Nachrichten und Lösungen

- Location
- supported-actions
- service-limits
- structure
- expression-syntax
- data-type
- referenced-data
- referenced-resource

`Location`

Ein Analyseergebnis mit Informationen zu `Location`entspricht der folgenden Fehlermeldung:

- Nachricht – Enthält zusätzliche Informationen über das Analyseergebnis. Dies können Informationen, Warnungen oder Fehlermeldungen sein.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie eine Aktion angegeben haben, die AWS IoT Events derzeit nicht unterstützt. Eine Liste der unterstützten Aktionen finden Sie unter Unterstützte Aktionen.

`supported-actions`

Ein Analyseergebnis mit Informationen zu `supported-actions`entspricht den folgenden Fehlermeldungen:

- Meldung: Ungültiger Aktionstyp in Aktionsdefinition vorhanden: *`Aktionsdefinition`* .

Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie eine Aktion angegeben haben, die AWS IoT Events derzeit nicht unterstützt. Eine Liste der unterstützten Aktionen finden Sie unter Unterstützte Aktionen.

- Message: DetectorModel definition hat eine *aws-service* Aktion, aber der *aws-service* Service wird in der Region *region-name* nicht unterstützt.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn die von Ihnen angegebene Aktion von unterstützt wirdAWS IoT Events, die Aktion jedoch in Ihrer aktuellen Region nicht verfügbar ist. Dies kann auftreten, wenn Sie versuchen, Daten an einen -AWSService zu senden, der in der Region nicht verfügbar ist. Sie müssen auch dieselbe Region für AWS IoT Events sowohl als auch für die AWS Services auswählen, die Sie verwenden.

`service-limits`

Ein Analyseergebnis mit Informationen zu `service-limits`entspricht den folgenden Fehlermeldungen:

- Nachricht: Der in der Nutzlast zulässige Inhaltsausdruck hat das Limit *content-expression-size* von Bytes im *Ereignisnamen* im *Statusstatusname* überschritten.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn der Inhaltsausdruck für Ihre Aktionsnutzlast größer als 1024 Byte ist. Die Größe des Inhaltsausdrucks für eine Nutzlast kann bis zu 1024 Byte betragen.
- Meldung: Die Anzahl der in der Detektormodelldefinition zulässigen Status hat das Limit überschritten*states-per-detector-model*.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Ihr Detektormodell mehr als 20 Status hat. Ein Detektormodell kann bis zu 20 Status haben.
- Meldung: Die Dauer für timer-*timer-name* sollte mindestens *minimum-timer-duration* Sekunden lang sein.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn die Dauer Ihres Timers weniger als 60 Sekunden beträgt. Wir empfehlen, dass die Dauer eines Timers zwischen 60 und 31622400 Sekunden liegt. Wenn Sie einen Ausdruck für die Dauer Ihres Timers angeben, wird das ausgewertete Ergebnis des Dauerausdrucks auf die nächste ganze Zahl abgerundet.
- Meldung: Anzahl der zulässigen Aktionen pro Ereignis hat das Limit *actions-per-event* in der Detektormodelldefinition überschritten

Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn das Ereignis mehr als 10 Aktionen umfasst. Sie können bis zu 10 Aktionen für jedes Ereignis in Ihrem Detektormodell haben.

- Meldung: Die Anzahl der zulässigen Übergangsereignisse pro Status hat das Limit *`transition-events-per-state`* in der Detektormodelldefinition überschritten.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn der Status mehr als 20 Übergangsereignisse enthält. Sie können bis zu 20 Übergangsereignisse für jeden Zustand in Ihrem Detektormodell haben.

- Meldung: Anzahl der zulässigen Ereignisse pro Status hat das Limit *`events-per-state`* in der Detektormodelldefinition überschritten

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn der Status mehr als 20 Ereignisse enthält. Sie können bis zu 20 Ereignisse für jeden Zustand in Ihrem Detektormodell haben.

- Meldung: Die maximale Anzahl von Detektormodellen, die einer einzelnen Eingabe zugeordnet werden können, hat möglicherweise das Limit erreicht. Input*`-Eingabename`* wird in *`detector-models-per-input`*Detektormodellrouten verwendet.

  Lösung: Sie erhalten diese Warnmeldung möglicherweise, wenn Sie versucht haben, eine Eingabe an mehr als 10 Detektormodelle weiterzuleiten. Sie können einem einzelnen Detektormodell bis zu 10 verschiedene Detektormodelle zuordnen.

`structure`

Ein Analyseergebnis mit Informationen zu `structure`entspricht den folgenden Fehlermeldungen:

- Meldung: Für Aktionen darf nur ein Typ definiert sein, es wurde jedoch eine Aktion mit -*`number-of-types`*Typen gefunden. Bitte teilen Sie sich in separate Aktionen auf.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie zwei oder mehr Aktionen in einem einzigen Feld angegeben haben, indem Sie API-Operationen zum Erstellen oder Aktualisieren Ihres Detektormodells verwenden. Sie können ein Array von -`Action`Objekten definieren. Stellen Sie sicher, dass Sie jede Aktion als separates Objekt definieren.

- Meldung: Der TransitionEvent *`transition-event-name`* wechselt in einen nicht vorhandenen *`Statusnamen`* .

Lösung: Möglicherweise erhalten Sie diese Fehlermeldung, wenn den nächsten Status, auf den Ihr Übergangsereignis verwiesen hat, nicht finden AWS IoT Events konnte. Stellen Sie sicher, dass der nächste Status definiert ist und dass Sie den richtigen Statusnamen eingegeben haben.

- Meldung: Der DetectorModelDefinition hatte einen gemeinsamen Statusnamen: *`State-Name`* mit *`number-of-states`* Wiederholungen gefunden.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie denselben Namen für einen oder mehrere Status verwenden. Stellen Sie sicher, dass Sie jedem Zustand in Ihrem Detektormodell einen eindeutigen Namen geben. Der Statusname muss 1–128 Zeichen lang sein. Gültige Zeichen: a-z, A-Z, 0-9, _ (Unterstrich) und - (Bindestrich).

- Meldung: Die der Definition initialStateName *`initial-state-name`* entspricht keinem definierten Zustand.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn der ursprüngliche Statusname falsch ist. Das Detektormodell bleibt im anfänglichen Zustand (Start), bis eine Eingabe eintrifft. Sobald eine Eingabe eintrifft, wechselt das Detektormodell sofort in den nächsten Zustand. Stellen Sie sicher, dass der anfängliche Statusname der Name eines definierten Status ist und dass Sie den richtigen Namen eingeben.

- Meldung: Detector Model Definition muss mindestens eine Eingabe in einer Bedingung verwenden.

  Lösung: Dieser Fehler wird möglicherweise angezeigt, wenn Sie keine Eingabe in einer Bedingung angegeben haben. Sie müssen mindestens eine Eingabe in mindestens einer Bedingung verwenden. Andernfalls AWS IoT Eventswertet keine eingehenden Daten aus.

- Meldung: Nur eine von Sekunden und durationExpression können in festgelegt werden SetTimer.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie sowohl als auch `seconds` `durationExpression` für Ihren Timer verwendet haben. Stellen Sie sicher, dass Sie entweder `seconds` oder `durationExpression` als Parameter von verwenden`SetTimerAction`. Weitere Informationen finden Sie unter SetTimerAction in der AWS IoT Events-API-Referenz.

- Meldung: Eine Aktion in Ihrem Detektormodell ist nicht erreichbar. Überprüfen Sie den Zustand, der die Aktion initiiert.

  Lösung: Wenn eine Aktion in Ihrem Detektormodell nicht erreichbar ist, wird die Bedingung des Ereignisses als falsch ausgewertet. Überprüfen Sie den Zustand des Ereignisses, das die Aktion enthält, um sicherzustellen, dass es als wahr ausgewertet wird. Wenn die Bedingung des Ereignisses „true“ ergibt, sollte die Aktion erreichbar werden.

- Meldung: Ein Eingabeattribut wird gelesen, dies kann jedoch durch einen Timer-Ablauf verursacht werden.

  Lösung: Der Wert eines Eingabeattributs kann gelesen werden, wenn einer der folgenden Fälle eintritt:

  - Ein neuer Eingabewert wurde empfangen.
  - Wenn ein Timer im Detektor abgelaufen ist.

  Um sicherzustellen, dass ein Eingabeattribut nur ausgewertet wird, wenn der neue Wert für diese Eingabe empfangen wird, fügen Sie wie folgt einen Aufruf der `triggerType(“Message”)` Funktion in Ihre Bedingung ein:

  Die ursprüngliche Bedingung, die im Detektormodell ausgewertet wird:

  ```
  if ($input.HeartBeat.status == “OFFLINE”)
  ```

  würde in etwa wie folgt aussehen:

  ```
  if ( triggerType("MESSAGE") &&  $input.HeartBeat.status == “OFFLINE”)
  ```

  wobei ein Aufruf der `triggerType(“Message”)` Funktion vor der ersten Eingabe in der Bedingung erfolgt. Mit dieser Technik wird die `triggerType("Message")` Funktion als wahr ausgewertet und erfüllt die Bedingung, einen neuen Eingabewert zu empfangen. Weitere Informationen zur Verwendung der `triggerType` Funktion finden Sie unter `triggerType` im Abschnitt Ausdrücke im -AWS IoT EventsEntwicklerhandbuch.
- Meldung: Ein Status in Ihrem Detektormodell ist nicht erreichbar. Überprüfen Sie die Bedingung, die zu einem Übergang in den gewünschten Zustand führt.

  Lösung: Wenn ein Zustand in Ihrem Detektormodell nicht erreichbar ist, wird eine Bedingung, die einen eingehenden Übergang in diesen Zustand verursacht, als falsch ausgewertet. Überprüfen Sie, ob die Bedingungen der eingehenden Übergänge in diesen nicht erreichbaren Zustand in Ihrem Detektormodell als wahr ausgewertet werden, sodass der gewünschte Zustand erreichbar werden kann.
- Nachricht: Ein ablaufender Timer kann dazu führen, dass eine unerwartete Anzahl von Nachrichten gesendet wird.

Lösung: Um zu verhindern, dass Ihr Detektormodell in einen unendlichen Zustand übergeht, um eine unerwartete Anzahl von Nachrichten zu senden, weil ein Timer abgelaufen ist, sollten Sie in den Bedingungen Ihres Detektormodells wie folgt einen Aufruf der `triggerType("Message")` Funktion verwenden:

Die ursprüngliche Bedingung, die im Detektormodell ausgewertet wird:

```
if (timeout("awake"))
```

würde in eine Bedingung umgewandelt werden, die der folgenden ähnelt:

```
 if (triggerType("MESSAGE") && timeout("awake"))
```

wobei ein Aufruf der `triggerType(“Message”)` Funktion vor der ersten Eingabe in der Bedingung erfolgt.

Diese Änderung verhindert, dass Timer-Aktionen in Ihrem Detektor ausgelöst werden, wodurch verhindert wird, dass eine unendliche Schleife von Nachrichten gesendet wird. Weitere Informationen zur Verwendung von Timer-Aktionen in Ihrem Detektor finden Sie auf der Seite Verwenden von integrierten Aktionen im -AWS IoT EventsEntwicklerhandbuch.

`expression-syntax`

Ein Analyseergebnis mit Informationen zu `expression-syntax`entspricht den folgenden Fehlermeldungen:

- Nachricht: Ihr Nutzlastausdruck {*`expression`* } ist ungültig. Der definierte Nutzlasttyp ist JSON, daher müssen Sie einen Ausdruck angeben, der zu einer Zeichenfolge ausgewertet AWS IoT Events wird.

  Lösung: Wenn der angegebene Nutzlasttyp JSON ist, prüft AWS IoT Events zunächst, ob der Service Ihren Ausdruck in eine Zeichenfolge auswerten kann. Das ausgewertete Ergebnis darf keine boolesche Zahl oder Zahl sein. Wenn die Validierung nicht erfolgreich ist, erhalten Sie diesen Fehler möglicherweise.
- Nachricht: `SetVariableAction.value` muss ein Ausdruck sein. Der Wert '*`variable-value`* ' konnte nicht analysiert werden

Lösung: Sie können verwenden`SetVariableAction`, um eine Variable mit einem `name` und zu definieren`value`. `value` kann eine Zeichenfolge, eine Zahl oder ein boolescher Wert sein. Sie können auch einen Ausdruck für die angeben`value`. Weitere Informationen finden Sie unter SetVariableActionin der APIAWS IoT Events-Referenz zu .

- Meldung: Wir konnten Ihren Ausdruck der Attribute (*`attribute-name`*) für die DynamoDB-Aktion nicht analysieren. DynamoDB Geben Sie den Ausdruck mit der richtigen Syntax ein.

  Lösung: Sie müssen Ausdrücke für alle Parameter in verwenden`DynamoDBAction`. - Ersetzungsvorlagen. Weitere Informationen finden Sie unter DynamoDBAction in der APIAWS IoT Events-Referenz zu .

- Meldung: Wir konnten Ihren Ausdruck des tableName für die DynamoDBv2-Aktion nicht analysieren. Geben Sie den Ausdruck mit der richtigen Syntax ein.

  Lösung: Der `tableName` in `DynamoDBv2Action` muss eine Zeichenfolge sein. Sie müssen einen Ausdruck für die verwenden`tableName`. Die Ausdrücke akzeptieren Literale, Operatoren, Funktionen, Referenzen und Substitutionsvorlagen. Weitere Informationen finden Sie unter DynamoDBv2Action in der APIAWS IoT Events-Referenz zu .

- Nachricht: Wir konnten Ihren Ausdruck nicht auf gültiges JSON auswerten. Die DynamoDBv2-Aktion unterstützt nur den JSON-Nutzlasttyp.

  Lösung: Der Nutzlasttyp für `DynamoDBv2` muss JSON sein. Stellen Sie sicher, dass Ihren Inhaltsausdruck für die Nutzlast auf gültiges JSON auswerten AWS IoT Events kann. Weitere Informationen finden Sie unter DynamoDBv2Action in der APIAWS IoT Events-Referenz zu .

- Nachricht: Wir konnten Ihren Inhaltsausdruck nicht für die Nutzlast des *`Aktionstyps`* analysieren. Geben Sie einen Inhaltsausdruck mit der richtigen Syntax ein.

  Lösung: Der Inhaltsausdruck kann Zeichenfolgen ('*`string`* '), Variablen ($variable.*`variable-name`*), Eingabewerte ($input.*`input-name`* .*`path-to-datum`*), Zeichenfolgenverkettungen und Zeichenfolgen enthalten, die enthalten`${}`.

- Meldung: Benutzerdefinierte Nutzlasten dürfen nicht leer sein.

  Lösung: Möglicherweise erhalten Sie diese Fehlermeldung, wenn Sie Benutzerdefinierte Nutzlast für Ihre Aktion ausgewählt und keinen Inhaltsausdruck in der AWS IoT Events Konsole eingegeben haben. Wenn Sie Benutzerdefinierte Nutzlast auswählen, müssen Sie unter Benutzerdefinierte Nutzlast einen Inhaltsausdruck eingeben. Weitere Informationen finden Sie unter Nutzlast in der APIAWS IoT Events-Referenz zu .

- Meldung: Der Dauerausdruck '*duration-expression* ' konnte nicht für den Timer '*timer-name* ' analysiert werden.

  Lösung: Das ausgewertete Ergebnis Ihres Dauerausdrucks für den Timer muss ein Wert zwischen 60 und 31622400 sein. Das ausgewertete Ergebnis der Dauer wird auf die nächste ganze Zahl abgerundet.
- Meldung: Ausdruck '*expression* ' für *action-name* konnte nicht analysiert werden

  Lösung: Sie erhalten diese Meldung möglicherweise, wenn der Ausdruck für die angegebene Aktion eine falsche Syntax aufweist. Stellen Sie sicher, dass Sie einen Ausdruck mit der richtigen Syntax eingeben. Weitere Informationen finden Sie unter Syntax.
- Nachricht: Ihr *fieldName* für `IotSitewiseAction` konnte nicht analysiert werden. Sie müssen die richtige Syntax in Ihrem Ausdruck verwenden.

  Lösung: Dieser Fehler wird möglicherweise angezeigt, wenn Ihren *fieldName* für nicht analysieren AWS IoT Events konnte`IotSitewiseAction`. Stellen Sie sicher, dass der *fieldName* einen Ausdruck verwendet, der analysieren AWS IoT Events kann. Weitere Informationen finden Sie unter IotSiteWiseAction in der AWS IoT Events-API-Referenz.

`data-type`

Ein Analyseergebnis mit Informationen zu `data-type`entspricht den folgenden Fehlermeldungen:

- Meldung: Der Dauerausdruck *duration-expression* für *timer-name* ist ungültig, er muss eine Zahl zurückgeben.

  Lösung: Möglicherweise erhalten Sie diese Fehlermeldung, wenn den Dauerausdruck für Ihren Timer nicht auf eine Zahl auswerten AWS IoT Events konnte. Stellen Sie sicher, dass Ihr in eine Zahl konvertiert werden `durationExpression` kann. Andere Datentypen, z. B. boolean, werden nicht unterstützt.
- Meldung: *Ausdrucksbedingungsausdruck* ist kein gültiger Bedingungsausdruck.

  Lösung: Möglicherweise erhalten Sie diese Fehlermeldung, wenn Ihre nicht `condition-expression` auf einen booleschen Wert auswerten AWS IoT Events konnte. Der boolesche Wert muss entweder `TRUE` oder sein`FALSE`. Stellen Sie sicher, dass Ihr Bedingungsausdruck in einen booleschen Wert konvertiert werden kann. Wenn es sich bei dem Ergebnis nicht um einen booleschen Wert handelt, entspricht er `FALSE` und ruft weder die Aktionen noch den im Ereignis `nextState` angegebenen auf.

- Meldung: Inkompatible Datentypen [*inferred-types*] zur *Referenz* im folgenden Ausdruck gefunden: *expression*

  Lösung: Lösung: Alle Ausdrücke für dasselbe Eingabeattribut oder dieselbe Variable im Detektormodell müssen auf denselben Datentyp verweisen.

  Verwenden Sie die folgenden Informationen, um das Problem zu beheben:

  - Wenn Sie eine Referenz als Operanden mit einem oder mehreren Operatoren verwenden, stellen Sie sicher, dass alle Datentypen, auf die Sie verweisen, kompatibel sind.

    Im folgenden Ausdruck 2 ist Ganzzahl beispielsweise ein Operand der && Operatoren == und . Um sicherzustellen, dass die Operanden kompatibel sind, `$variable.testVariable` müssen `$variable.testVariable + 1` sie auf eine Ganzzahl oder Dezimalzahl verweisen.

    Darüber hinaus 1 ist Ganzzahl ein Operand des -+Operators. Daher `$variable.testVariable` muss auf eine Ganzzahl oder Dezimalzahl verweisen.

    ```
    ‘$variable.testVariable + 1 == 2 && $variable.testVariable’
    ```

  - Wenn Sie eine Referenz als Argument verwenden, das an eine Funktion übergeben wird, stellen Sie sicher, dass die Funktion die Datentypen unterstützt, auf die Sie verweisen.

    Die folgende `timeout("`*time-name*`")` Funktion erfordert beispielsweise eine Zeichenfolge mit doppelten Anführungszeichen als Argument. Wenn Sie eine Referenz für den Wert *timer-name* verwenden, müssen Sie eine Zeichenfolge mit doppelten Anführungszeichen referenzieren.

    ```
    timeout("timer-name")
    ```

    > **Note**
    >
    > Wenn Sie für die `convert(`*type*`, `*expression*`)` Funktion eine Referenz für den *Typwert* verwenden, muss das ausgewertete Ergebnis Ihrer Referenz `String`, `Decimal`, oder sein`Boolean`.

  Weitere Informationen finden Sie unter Referenzen.

- Meldung: Inkompatible Datentypen [*inferred-types*], die mit *der Referenz* verwendet werden. Dies kann zu einem Laufzeitfehler führen.

Lösung: Sie erhalten diese Warnmeldung möglicherweise, wenn zwei Ausdrücke für dasselbe Eingabeattribut oder dieselbe Variable auf zwei Datentypen verweisen. Stellen Sie sicher, dass Ihre Ausdrücke für dasselbe Eingabeattribut oder dieselbe Variable im Detektormodell auf denselben Datentyp verweisen.

- Nachricht: Die Datentypen [*inferated-types],* die Sie für den Operator [*operator*] eingegeben haben, sind für den folgenden Ausdruck nicht kompatibel: '*expression* '

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Ihr Ausdruck Datentypen kombiniert, die nicht mit einem angegebenen Operator kompatibel sind. Im folgenden Ausdruck + ist der Operator beispielsweise mit den Datentypen Ganzzahl, Dezimal und Zeichenfolge kompatibel, jedoch nicht mit Operanden des booleschen Datentyps.

  ```
  true + false
  ```

  Sie müssen sicherstellen, dass die Datentypen, die Sie mit einem -Operator verwenden, kompatibel sind.

- Meldung: Die für *input-attribute* gefundenen Datentypen [*inferated-types]* sind nicht kompatibel und können zu einem Laufzeitfehler führen.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn zwei Ausdrücke für dasselbe Eingabeattribut auf zwei Datentypen verweisen, entweder für den `OnEnterLifecycle` eines -Zustands oder sowohl für den - `OnInputLifecycle` als auch `OnExitLifecycle` für einen -Zustand. Stellen Sie sicher, dass Ihre Ausdrücke in `OnEnterLifecycle` (oder sowohl `OnInputLifecycle` als auch `OnExitLifecycle`) für jeden Status Ihres Detektormodells auf denselben Datentyp verweisen.

- Nachricht: Der Nutzlastausdruck [*Ausdruck*] ist ungültig. Geben Sie einen Ausdruck an, der zur Laufzeit zu einer Zeichenfolge ausgewertet wird, da der Nutzlasttyp das JSON-Format hat.

  Lösung: Dieser Fehler wird möglicherweise angezeigt, wenn Ihr angegebener Nutzlasttyp JSON ist, aber seinen Ausdruck nicht als Zeichenfolge auswerten AWS IoT Events kann. Stellen Sie sicher, dass das ausgewertete Ergebnis eine Zeichenfolge ist, kein boolescher Wert oder eine Zahl.

- Nachricht: Ihr interpolierter Ausdruck {*interpolated-expression*} muss zur Laufzeit entweder zu einer Ganzzahl oder zu einem booleschen Wert ausgewertet werden. Andernfalls ist Ihr Nutzlastausdruck {*payload-expression*} zur Laufzeit nicht als gültiges JSON parsbar.

Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Ihren interpolierten Ausdruck nicht auf eine Ganzzahl oder einen booleschen Wert auswerten AWS IoT Events konnte. Stellen Sie sicher, dass Ihr interpolierter Ausdruck in eine Ganzzahl oder einen booleschen Wert konvertiert werden kann, da andere Datentypen, wie z. B. Tring, nicht unterstützt werden.

- Meldung: Der Ausdruckstyp im `IotSitewiseAction` *`Feldausdruck`* ist als Typ *`defined-type`* definiert und als Typ *`inferated-type`* abgeleitet. Der definierte Typ und der abgeleitete Typ müssen identisch sein.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Ihr Ausdruck in `propertyValue` der einen anderen Datentyp definiert `IotSitewiseAction` hat als der von abgeleitete DatentypAWS IoT Events. Stellen Sie sicher, dass Sie denselben Datentyp für alle Instances dieses Ausdrucks in Ihrem Detektormodell verwenden.

- Nachricht: Die für die `setTimer` Aktion verwendeten Datentypen [*`inferred-types`*] werden `Integer` für den folgenden Ausdruck nicht als ausgewertet: *`expression`*

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn der abgeleitete Datentyp für Ihren Dauerausdruck nicht Ganzzahl oder Dezimal ist. Stellen Sie sicher, dass Ihr in eine Zahl konvertiert werden `durationExpression` kann. Andere Datentypen wie Boolean und String werden nicht unterstützt.

- Nachricht: Die Datentypen [*`inferred-types`*] , die mit Operanden des Vergleichsoperators [*`operator`*] verwendet werden, sind im folgenden Ausdruck nicht kompatibel: *`expression`*

  Lösung: Die abgeleiteten Datentypen für die Operanden des *`Operators`* im bedingten Ausdruck (*`Ausdruck`*) Ihres Detektormodells stimmen nicht überein. Die Operanden müssen mit den übereinstimmenden Datentypen in allen anderen Teilen Ihres Detektormodells verwendet werden.

> **Tip**
>
> Sie können verwendenconvert, um den Datentyp eines Ausdrucks in Ihrem Detektormodell zu ändern. Weitere Informationen finden Sie unter Funktionen.

`referenced-data`

Ein Analyseergebnis mit Informationen zu `referenced-data`entspricht den folgenden Fehlermeldungen:

- Meldung: Erkannter defekter Timer: Timer-*Timer-Name* wird in einem Ausdruck verwendet, aber nie festgelegt.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie einen Timer verwenden, der nicht festgelegt ist. Sie müssen einen Timer festlegen, bevor Sie ihn in einem Ausdruck verwenden. Stellen Sie außerdem sicher, dass Sie den richtigen Timernamen eingeben.

- Meldung: Erkannte fehlerhafte Variable: *variable-name* wird in einem Ausdruck verwendet, aber nie festgelegt.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie eine Variable verwenden, die nicht festgelegt ist. Sie müssen eine Variable festlegen, bevor Sie sie in einem Ausdruck verwenden. Stellen Sie außerdem sicher, dass Sie den richtigen Variablennamen eingeben.

- Meldung: Erkannte fehlerhafte Variable: Eine Variable wird in einem Ausdruck verwendet, bevor sie auf einen Wert festgelegt wird.

  Lösung: Jede Variable muss einem Wert zugewiesen werden, bevor sie in einem Ausdruck ausgewertet werden kann. Legen Sie den Wert der Variablen vor jeder Verwendung fest, damit ihr Wert abgerufen werden kann. Stellen Sie außerdem sicher, dass Sie den richtigen Variablennamen eingeben.

`referenced-resource`

Ein Analyseergebnis mit Informationen zu `referenced-resource`entspricht den folgenden Fehlermeldungen:

- Meldung: Detector Model Definition enthält einen Verweis auf Input, der nicht existiert.

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Sie Ausdrücke verwenden, um auf eine nicht vorhandene Eingabe zu verweisen. Stellen Sie sicher, dass Ihr Ausdruck auf eine vorhandene Eingabe verweist, und geben Sie den richtigen Eingabenamen ein. Wenn Sie keine Eingabe haben, erstellen Sie zuerst eine.

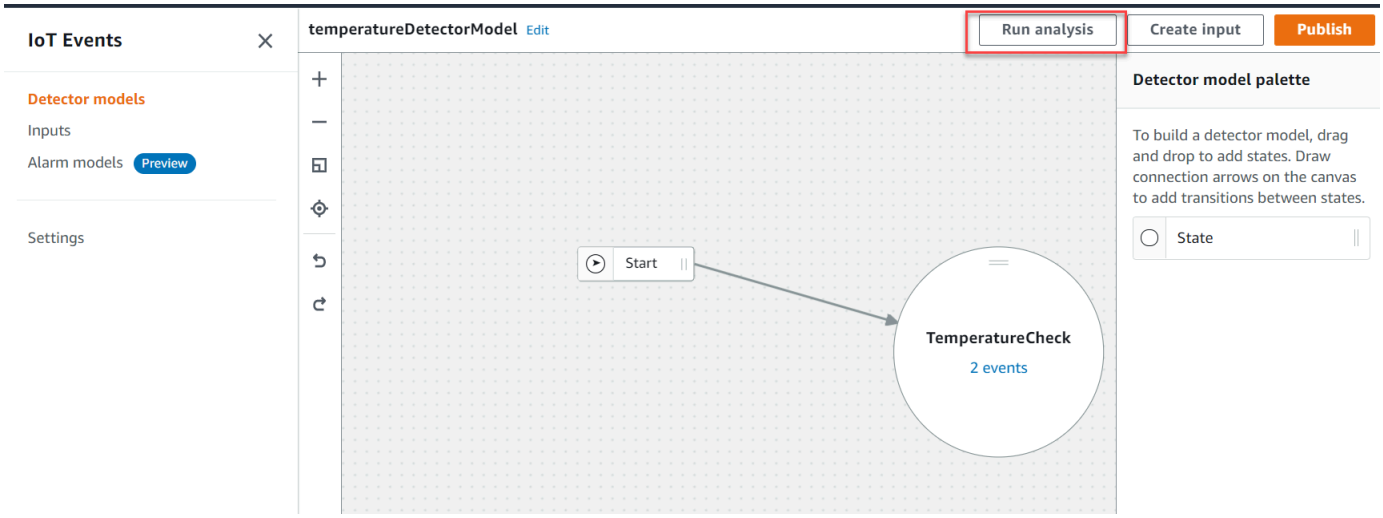
- Meldung: Detector Model Definition enthält ungültige InputName: *input-name*

  Lösung: Sie erhalten diese Fehlermeldung möglicherweise, wenn Ihr Detektormodell einen ungültigen Eingabenamen enthält. Stellen Sie sicher, dass Sie den richtigen Eingabenamen eingeben. Der Eingabename muss 1–128 Zeichen lang sein. Gültige Zeichen: a-z, A-Z, 0-9, _ (Unterstrich) und - (Bindestrich).

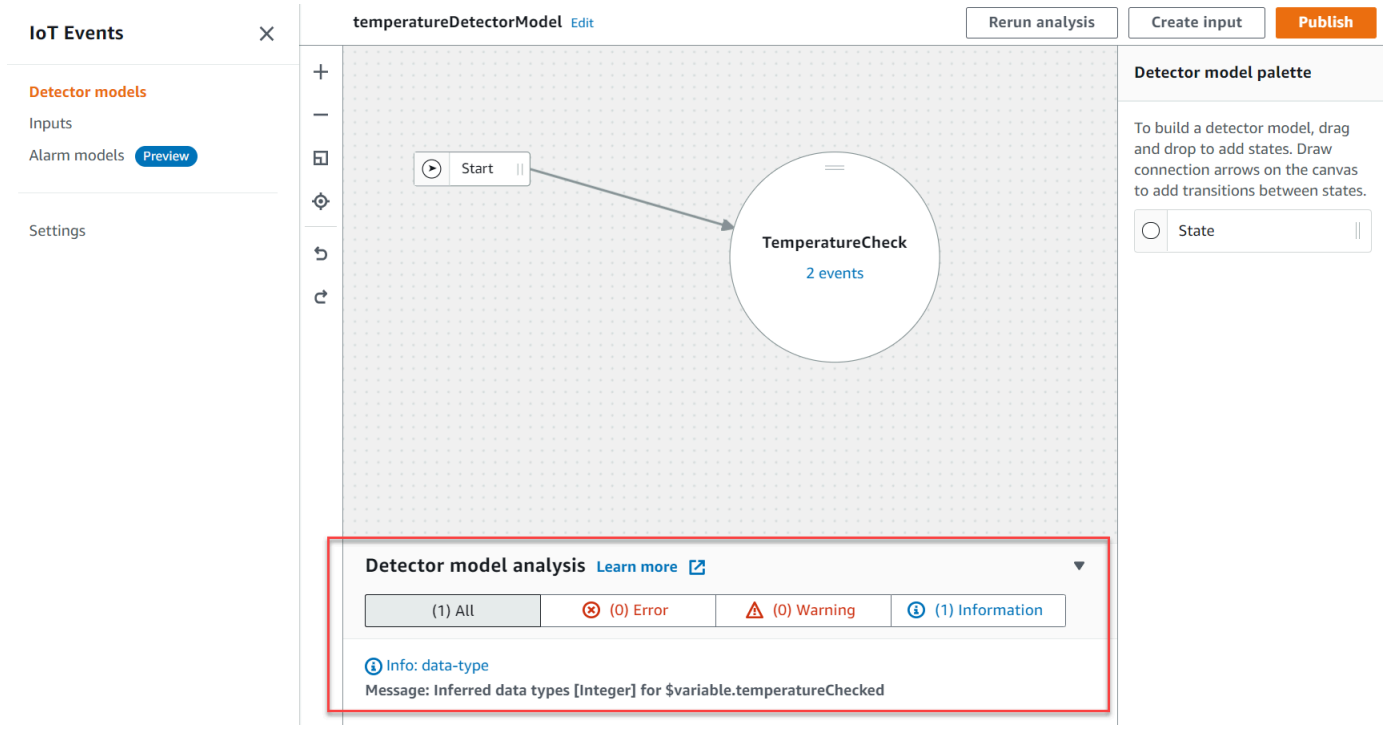
Analysieren eines Detektormodells (Konsole)

In den folgenden Schritten wird die AWS IoT Events Konsole verwendet, um ein Detektormodell zu analysieren.

1. Melden Sie sich an der AWS IoT Events-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Detektormodelle aus.
3. Wählen Sie unter Detektormodelle das Zieldetektormodell aus.
4. Wählen Sie auf der Seite mit dem Meldermodell die Option Bearbeiten aus.
5. Wählen Sie in der oberen rechten Ecke die Option Analyse ausführen aus.



Im Folgenden finden Sie ein Beispiel für ein Analyseergebnis in der AWS IoT Events Konsole.



> **Note**
>
> Nachdem Sie AWS IoT Events mit der Analyse Ihres Detektormodells begonnen haben, haben Sie bis zu 24 Stunden Zeit, um die Analyseergebnisse abzurufen.

Analysieren eines Detektormodells (AWS CLI)

In den folgenden Schritten wird AWS CLI ein Detektormodell analysiert.

1. Führen Sie den folgenden Befehl aus, um eine Analyse zu starten.

```
aws iotevents start-detector-model-analysis --cli-input-json file://file-name.json
```

> **Note**
>
> Ersetzen Sie *file-name* durch den Namen der Datei, die die Definition des Detektormodells enthält.

Example Definition des Detektormodells

```
{
    "detectorModelDefinition": {
        "states": [
            {
                "stateName": "TemperatureCheck",
                "onInput": {
                    "events": [
                        {
                            "eventName": "Temperature Received",
                            "condition":
 "isNull($input.TemperatureInput.sensorData.temperature)==false",
                            "actions": [
                                {
                                    "iotTopicPublish": {
                                        "mqttTopic": "IoTEvents/Output"
                                    }
                                }
                            ]
                        }
                    ],
                    "transitionEvents": []
                },
                "onEnter": {
                    "events": [
                        {
                            "eventName": "Init",
                            "condition": "true",
                            "actions": [
                                {
                                    "setVariable": {
                                        "variableName": "temperatureChecked",
                                        "value": "0"
                                    }
                                }
                            ]
                        }
                    ]
                },
                "onExit": {
                    "events": []
```

```
                }
            }
        ],
        "initialStateName": "TemperatureCheck"
    }
}
```

Wenn Sie den verwendenAWS CLI, um ein vorhandenes Detektormodell zu analysieren, wählen Sie eine der folgenden Optionen, um die Definition des Detektormodells abzurufen:

- Wenn Sie die AWS IoT Events Konsole verwenden möchten, gehen Sie wie folgt vor:
  1. Wählen Sie im Navigationsbereich die Option Detector models aus.
  2. Wählen Sie unter Detektormodelle das Zieldetektormodell aus.
  3. Wählen Sie unter Aktion die Option Detektormodell exportieren aus, um das Detektormodell herunterzuladen. Das Detektormodell wird in JSON gespeichert.
  4. Öffnen Sie die JSON-Datei für das Detektormodell.
  5. Sie benötigen nur das `detectorModelDefinition` Objekt. Entfernen Sie Folgendes:
     - Die erste geschweifte Klammer ({) oben auf der Seite
     - Die Linie `detectorModel`
     - Das `detectorModelConfiguration` Objekt
     - Die letzte geschweifte Klammer (}) unten auf der Seite
  6. Speichern Sie die Datei.
- Wenn Sie den verwenden möchtenAWS CLI, gehen Sie wie folgt vor:
  1. Führen Sie folgenden Befehl von einem Terminal aus.

     ```
     aws iotevents describe-detector-model --detector-model-name detector-model-name
     ```

  2. *`detector-model-name`*Ersetzen Sie es durch den Namen Ihres Meldermodells.
  3. Kopieren Sie das `detectorModelDefinition` Objekt in einen Texteditor.
  4. Fügen Sie geschweifte Klammern ({}) außerhalb von hinzu. `detectorModelDefinition`
  5. Speichern Sie die Datei in JSON.

Example Beispielantwort

```
{
```

```
    "analysisId": "c1133390-14e3-4204-9a66-31efd92a4fed"
}
```

2. Kopieren Sie die Analyse-ID aus der Ausgabe.
3. Führen Sie den folgenden Befehl aus, um den Status der Analyse abzurufen.

```
aws iotevents describe-detector-model-analysis --analysis-id "analysis-id"
```

> **Note**
>
> Ersetzen Sie *analysis-id durch die Analyse-ID*, die Sie kopiert haben.

Example Beispielantwort

```
{
    "status": "COMPLETE"
}
```

Der Status kann einer der folgenden Werte sein:

- `RUNNING`— analysiert Ihr AWS IoT Events Detektormodell. Dieser Vorgang kann bis zu einer Minute dauern.
- `COMPLETE`— die Analyse Ihres Detektormodells AWS IoT Events abgeschlossen.
- `FAILED`— AWS IoT Events konnte Ihr Detektormodell nicht analysieren. Bitte versuchen Sie es später erneut.

4. Führen Sie den folgenden Befehl aus, um ein oder mehrere Analyseergebnisse des Detektormodells abzurufen.

> **Note**
>
> Ersetzen Sie *analysis-id durch die Analyse-ID*, die Sie kopiert haben.

```
aws iotevents get-detector-model-analysis-results --analysis-id "analysis-id"
```

Example Beispielantwort

```
{
    "analysisResults": [
        {
            "type": "data-type",
            "level": "INFO",
            "message": "Inferred data types [Integer] for
 $variable.temperatureChecked",
            "locations": []
        },
        {
            "type": "referenced-resource",
            "level": "ERROR",
            "message": "Detector Model Definition contains reference to Input
 'TemperatureInput' that does not exist.",
            "locations": [
                {
                    "path": "states[0].onInput.events[0]"
                }
            ]
        }
    ]
}
```

> **Note**
>
> Nachdem Sie AWS IoT Events mit der Analyse Ihres Detektormodells begonnen haben, haben Sie bis zu 24 Stunden Zeit, um die Analyseergebnisse abzurufen.

AWS IoT Events-Befehle

In diesem Kapitel finden Sie detaillierte Informationen zu allen API-Vorgängen, einschließlich Beispielanfragen, Antworten und Fehlern für die unterstützten Webdienstprotokolle. AWS IoT Events

AWS IoT Events-Aktionen

Sie können AWS IoT Events API-Befehle verwenden, um Eingaben und Detektormodelle zu erstellen, zu lesen, zu aktualisieren und zu löschen und deren Versionen aufzulisten. Weitere Informationen finden Sie AWS IoT Events in der AWS IoT EventsAPI-Referenz unter Aktionen und Datentypen, die von unterstützt werden.

Die AWS IoT EventsAbschnitte in der AWS CLIBefehlsreferenz enthalten die AWS CLI Befehle, die Sie zur Verwaltung und Bearbeitung AWS IoT Events verwenden können.

AWS IoT Events-Daten

Sie können die AWS IoT Events Daten-API-Befehle verwenden, um Eingaben an Melder zu senden, Melder aufzulisten und den Status eines Melders anzuzeigen oder zu aktualisieren. Weitere Informationen finden Sie in der AWS IoT EventsAPI-Referenz unter den Aktionen und Datentypen, die von AWS IoT Events Data unterstützt werden.

Die AWS IoT EventsDatenabschnitte in der AWS CLIBefehlsreferenz enthalten die AWS CLI Befehle, mit denen Sie AWS IoT Events Daten verarbeiten können.

Dokumentverlauf

In der folgenden Tabelle werden die wichtigen Änderungen am AWS IoT EventsEntwicklerhandbuch nach dem 17. September 2020 beschrieben. Für weitere Informationen zu Aktualisierungen dieser Dokumentation können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Start der Region	AWS IoT Events ist jetzt in der Region Asien-Pazifik (Mumbai) erhältlich.	30. September 2021
Start der Region	AWS IoT Eventsist jetzt in der Region AWS GovCloud (USA West) verfügbar.	22. September 2021
Beheben Sie Fehler bei einem Detektormodell, indem Sie Analysen ausführen	AWS IoT Eventskann jetzt Ihr Detektormodell analysier en und Analyseergebnisse generieren, die Sie zur Fehlerbehebung bei Ihrem Detektormodell verwenden können.	23. Februar 2021
Start in der Region	AWS IoT EventsIn China (Peking) eingeführt.	30. September 2020
Verwendung von Ausdrücken	Es wurden Beispiele hinzugefü gt, die Ihnen zeigen, wie man Ausdrücke schreibt.	22. September 2020
Überwachung mit Alarmen	Alarme helfen Ihnen dabei, Ihre Daten auf Änderunge n zu überwachen. Sie können Alarme erstellen, die Benachrichtigungen senden,	1. Juni 2020

	wenn ein Schwellenwert überschritten wird.

Frühere Aktualisierungen

In der folgenden Tabelle werden wichtige Änderungen am AWS IoT EventsEntwicklerhandbuch vor dem 18. September 2020 beschrieben.

Änderung	Beschreibung	Datum
Ergänzungen	Typvalidierung zu hinzugefügtReferenzen.	3. August 2020
Ergänzungen	Regionsinformationen wurden hinzugefügt zuArbeiten mit anderen - AWS Services.	7. Mai 2020
Ergänzungen, Aktualisierungen	Die Funktion „Payload Customization“ und neue Ereignisaktionen wurden hinzugefügt: Amazon DynamoDB und. AWS IoT SiteWise	27. April 2020
Bearbeitungen	Neue Beschreibungen von Zustandsmaschinen-Konzepten hinzugefügt. Allgemeine Bearbeitung von Inhalten.	31. Oktober 2019
Ergänzungen	Neue integrierte Funktionen für bedingte Ausdrücke von Detektormodellen hinzugefügt.	10. September 2019
Ergänzungen	Beispiele für Detektormodelle hinzugefügt.	5. August 2019

Änderung	Beschreibung	Datum
Ergänzungen	Neue Ereignisaktionen hinzugefügt: Lambda, Amazon SQS, Kinesis Data Firehose und Eingabe. AWS IoT Events	19. Juli 2019
Ergänzungen, Korrekturen	Beschreibung der `timeout()` Funktion korrigiert. Bewährte Methode in Bezug auf Kontoinaktivität hinzugefügt.	11. Juni 2019
Korrekturen	Aktualisiert: Seitenbild mit Debug-Optionen für die Konsole; Richtlinie für Konsolenberechtigungen.	5. Juni 2019
Aktualisierungen	AWS IoT EventsDer Dienst ist jetzt allgemein verfügbar.	30. Mai 2019
Ergänzungen, Aktualisierungen	Aktualisierte Sicherheitsinformationen; Ein Beispiel für ein kommentiertes Meldermodell wurde hinzugefügt.	22. Mai 2019
Korrekturen	Aktualisiert: Link zum eingeschränkten Vorschau-Download; Beispiele für Amazon SNS SNS-Nutzdaten; Ergänzungen zu den erforderlichen Berechtigungen für. CreateDetectorModel	17. Mai 2019
Ergänzungen	Informationen zur Sicherheit hinzugefügt.	9. Mai 2019

Änderung	Beschreibung	Datum
Korrekturen	Link zum eingeschränkten Vorschau-Download korrigiert.	19. April 2019
Bearbeitungen	Redaktionelle Verbesserungen.	16. April 2019
Limitierte Vorschauversion	Eingeschränkte Vorschauversion der Dokumentation.	28. März 2019
Bearbeitungen	Redaktionelle Verbesserungen.	18. Mai 2018

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.