Amazon IoT Events

开发人员指南

Amazon IoT Events: 开发人员指南

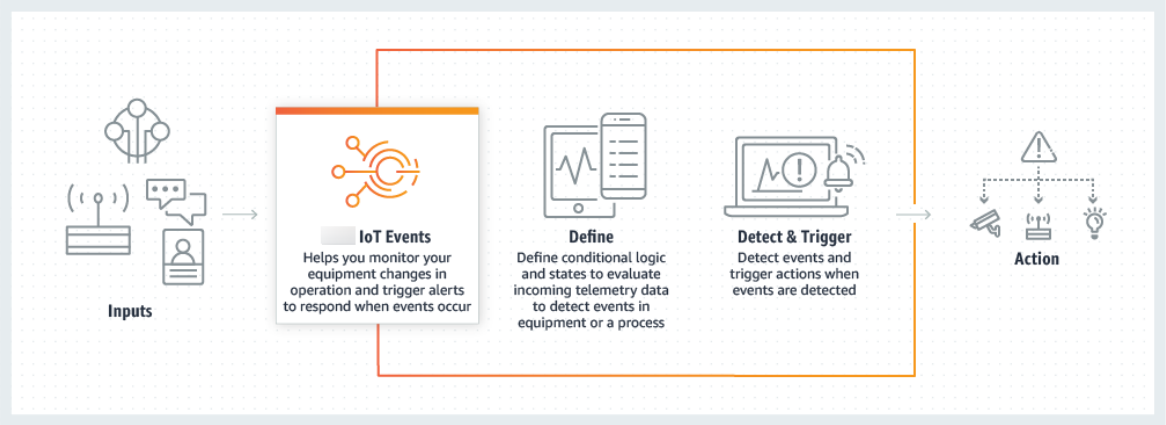
Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅中国的 Amazon Web Services 服务入门。

Table of Contents

什么是 Amazon IoT Events？

Amazon IoT Events让您了解如何监控您的设备和设备机群中的故障情况或操作中的更改，并在发生此类事件时触发措施。Amazon IoT Events持续监视来自设备、进程、应用程序和其他的 IoT 传感器数据Amazon用于识别重大事件的服务，以便您采取行动。

您可以使用Amazon IoT Events在中构建复杂的事件监控应用程序Amazon您可以通过以下方式访问的云Amazon IoT Events控制台或 API。



优点和特点

接受来自多个来源的输入

Amazon IoT Events接受来自许多 IoT 遥测数据源的输入。其中包括传感器设备、管理应用程序和其他Amazon IoT服务，例如Amazon IoT Core和Amazon IoT Analytics. 您可以将任何遥测数据输入推送到Amazon IoT Events通过使用标准 API 接口 (`BatchPutMessage`API) 。

使用简单逻辑表达式识别复杂的事件模式

Amazon IoT Events可以识别涉及来自单个 IoT 设备或应用程序或来自不同设备和许多独立传感器的多个输入的事件模式。这特别有用，因为每个传感器和应用程序都提供重要信息。但是，只有将不同的传感器和应用程序数据相结合，您才能全面了解操作的性能和质量。您可以配置Amazon IoT Events探测器使用简单的逻辑表达式而不是复杂的代码来识别这些事件。

根据事件触发操作

Amazon IoT Events让您了解如何在 Amazon SS (Amazon SS) 中直接触发措施，Amazon IoT Core、Lambda、Amazon SQS 和Amazon Kinesis Firehose。您还可以触发Amazon Lambda函数，使用Amazon IoT规则引擎，可使用其他服务（例如 Amazon Connect 或您自己的企业资源规划 (ERP) 应用程序）采取措施。

Amazon IoT Events包括您可以执行的预建操作库，还允许您定义自己的操作。

自动扩展以满足您的机队需求

Amazon IoT Events当您连接同构设备时，会自动扩展。您可以为特定类型的设备定义一次检测器，该服务将自动扩展和管理该设备连接到的所有实例Amazon IoT Events.

使用案例

监控和维护远程设备

您需要监控一组远程部署的计算机。如果一个处理器停止运行，并且您没有其他背景信息来了解导致故障的原因，则可能必须立即更换整个处理单元或机器。但这是不可持续的。与Amazon IoT Events您可以接收来自每台计算机上的多个传感器的消息，并使用一段时间内发送的错误代码来诊断确切的问题。现在，您无需更换所有部件，而是获得了派遣技术人员所需的信息，只需要更换需要更换的部件。有了数百万台机器，可以节省多达数百万美元，从而降低拥有或维护每台机器的总成本。

管理工业机器人

您在设施内部署机器人以自动移动包裹。为了将机器人的成本降至最低，机器人配备了简单、低成本的传感器，可以向云端报告信息。但是您的机器人有数十个传感器和数百种操作模式，因此很难在问题发生时检测出来。使用Amazon IoT Events，您可以构建一个专家系统来处理云中的传感器数据，并创建警报，以便在即将发生故障时自动警告技术人员。

轨道楼宇自动化系统

您运营的大量数据中心必须对其进行高温和低湿度监控，以防止在突破这些环境阈值时发生设备故障。您使用的传感器是从许多制造商那里购买的，每种类型都有自己的管理软件。但是，来自不同供应商的管理软件不兼容，因此很难发现问题。使用Amazon IoT Events，您可以设置警报，在发生故障之前就将供暖和冷却系统出现问题通知运营分析师。通过这种方式，您可以防止数据中心意外关闭，这会导致数千美元的设备更换和潜在的收入损失。

设置 Amazon IoT Events

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

设置 Amazon IoT Events 的权限

本部分介绍使用某些功能所需的角色和权限Amazon IoT Events. 您可以使用Amazon CLI命令或Amazon Identity and Access Management(IAM) 控制台，用于创建角色和关联权限策略以访问资源或执行特定功能Amazon IoT Events.

这些区域有：IAM 用户指南提供了有关安全控制访问权限的更多详细信息Amazon资源。ook er 有关以下内容的特定信息Amazon IoT Events请参阅的操作、资源和条件键Amazon IoT Events.

操作权限

Amazon IoT Events使您能够触发使用其他操作的操作Amazon服务 为此，您必须授予Amazon IoT Events代表您向您提供这些操作的工作流程权限。本部分包含操作列表和示例策略，该策略授予对您的资源执行所有这些操作的权限。更改*##*和*## ID*根据需要提供参考资料。如果可能，您还应更改通配符 (*)，使其指代将要访问的特定资源。您可以使用 IAM 控制台向以下内容授予权限Amazon IoT Events发送您定义的 Amazon SNS 警报。有关更多信息，请参阅 使用 IAM 控制台管理角色和权限 (p. 5)。

Amazon IoT Events支持以下允许您使用计时器或设置变量的操作：

- `setTimer` (p. 87)创建计时器。
- `resetTimer` (p. 88)重置计时器。
- `clearTimer` (p. 88)删除计时器。
- `setVariable` (p. 88)创建变量。

Amazon IoT Events支持下列操作，可让您使用Amazon服务

- `iotTopicPublish` (p. 89)在 MQTT 主题上发布消息。
- `iotEvents` (p. 90)将数据发送到Amazon IoT Events作为输入值。
- `iotSiteWise` (p. 90) – 将数据发送到 Amazon IoT SiteWise 中的资产属性。
- `dynamoDB` (p. 92)将数据发送到 Amazon DynamoDB 表。
- `dynamoDBv2` (p. 93)将数据发送到 Amazon DynamoDB 表。
- `firehose` (p. 94)将数据发送到 Amazon Kinesis Data Firehose 流。
- `lambda` (p. 94)调用一个Amazon Lambda函数。
- `sns` (p. 95)以推送通知的形式发送数据。
- `sqs` (p. 95)将数据发送到 Amazon SQS 队列。

Example 策略

```
{
   "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>:*"
     }
   ]
}
```

保护输入数据

重要的是要考虑谁可以授予在探测器模型中使用的输入数据的访问权限。如果您要限制某个用户或实体的总体权限，但允许其创建或更新探测器模型，则还必须授予该用户或实体更新输入路由的权限。这意味着，除了授予权限外`iotevents:CreateDetectorModel`和`iotevents:UpdateDetectorModel`，您还必须授予权限`iotevents:UpdateInputRouting`.

Example

以下策略添加了对的权限`iotevents:UpdateInputRouting`.

```
{
```

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

您可以指定输入Amazon Resource Name (ARN)” 列表，而不是通配符”*“为了”Resource“将此权限限制为特定输入。这使您可以限制对用户或实体创建或更新的探测器模型所消耗的输入数据的访问。

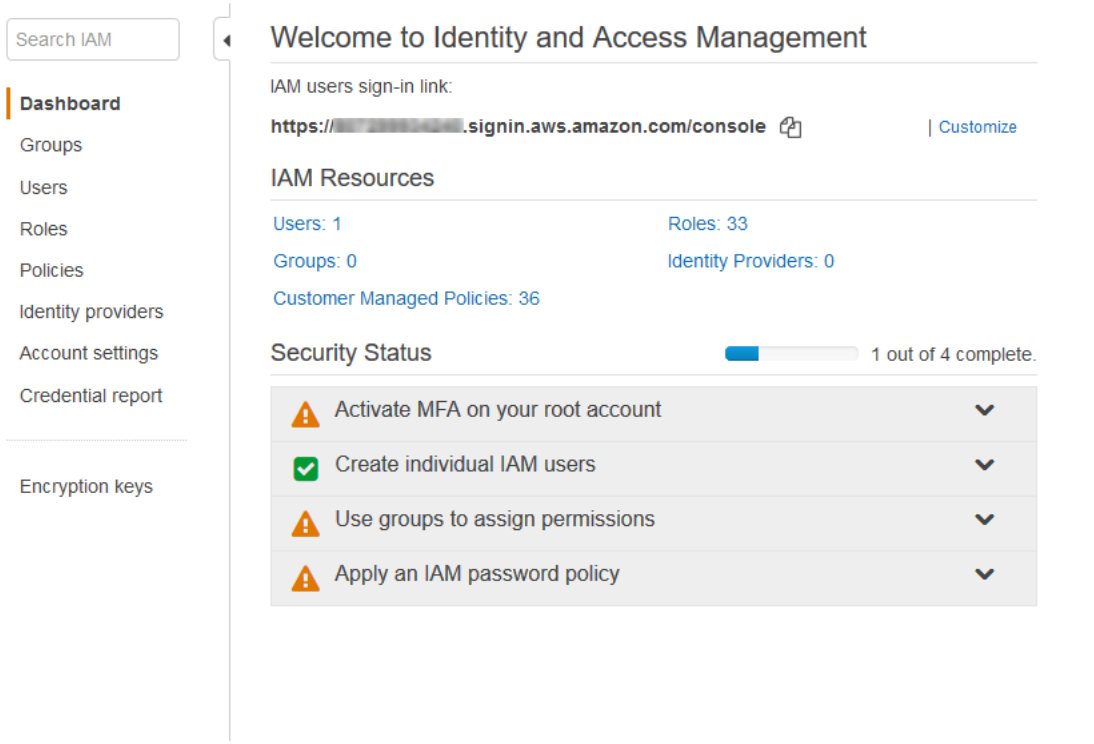
使用 IAM 控制台管理角色和权限

以下示例说明如何使用 IAM 控制台向以下对象授予权限Amazon IoT Events代表您生成 Amazon SNS 提醒。您还可以将角色附加到用于定义和发布的任何实体（用户或账户所有者）Amazon IoT Events检测器模型，包含发送 Amazon SNS 警报的事件操作。

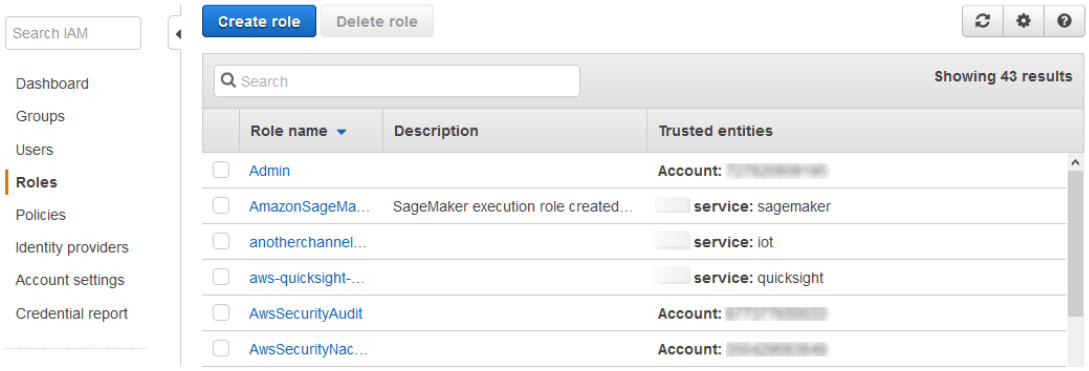
要完成此示例，您需要现有的 ARN。Amazon SNS 主题在中使用Amazon IoT Events.

使用 IAM 控制台管理角色和权限

1. 登录Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择面板板,角色.



3. 选择Create role（创建角色）。

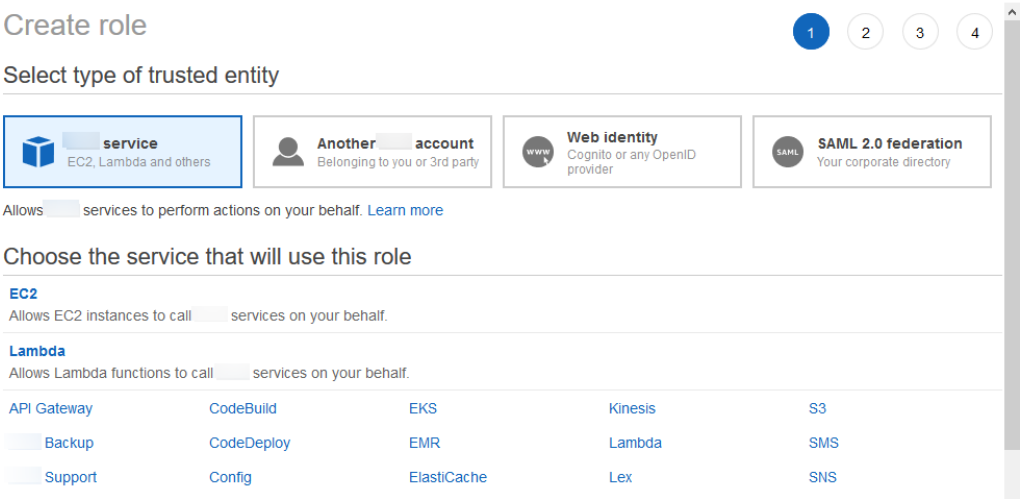


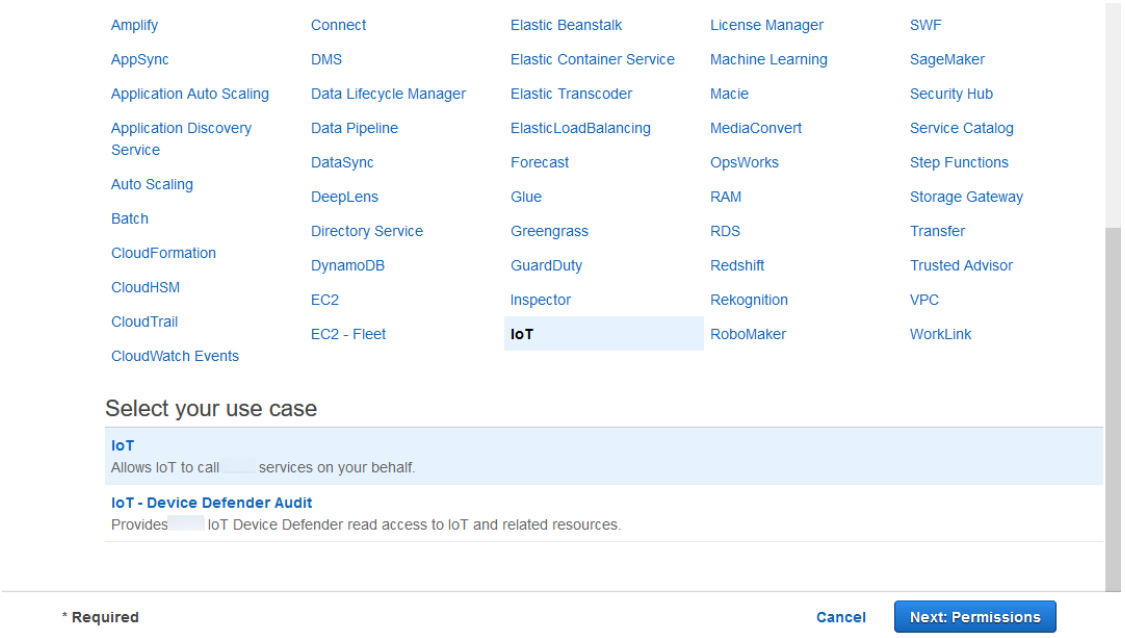
4. 在创建角色页面，请执行以下操作。

   a. 对于Select type of trusted entity（选择受信任实体的类型），选择 Amazon service（服务）。

   b. 对于选择将使用此角色的服务，选择IoT.

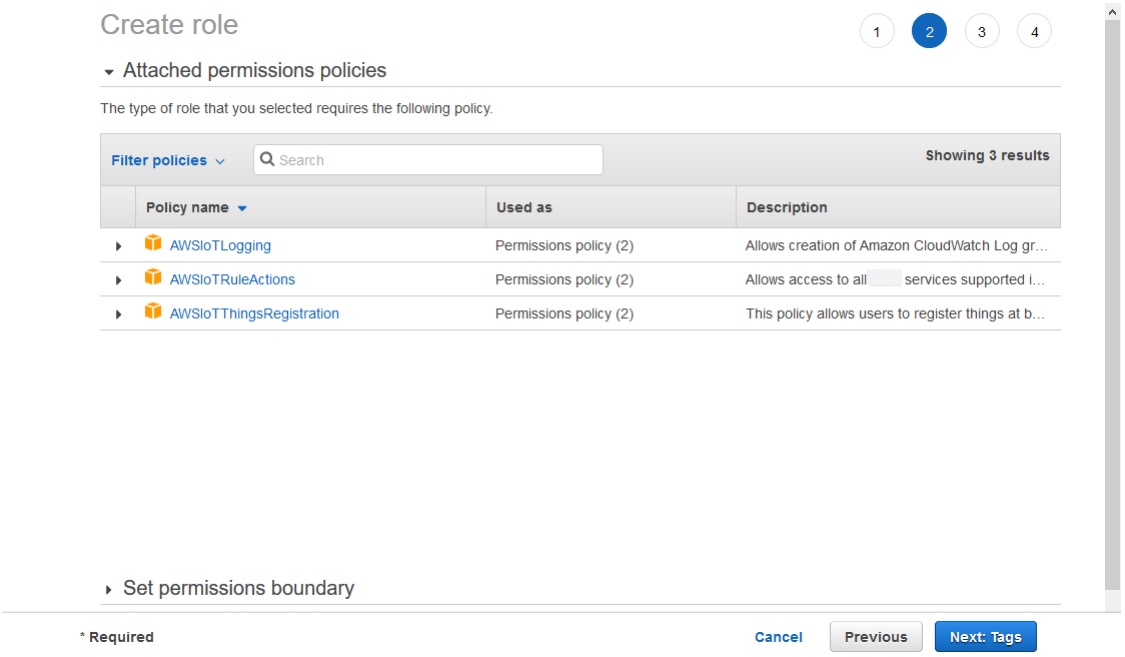
   c. 对于选择您的使用案例，选择IoT然后选择下一步:s（Permissions (下一步：权限)。



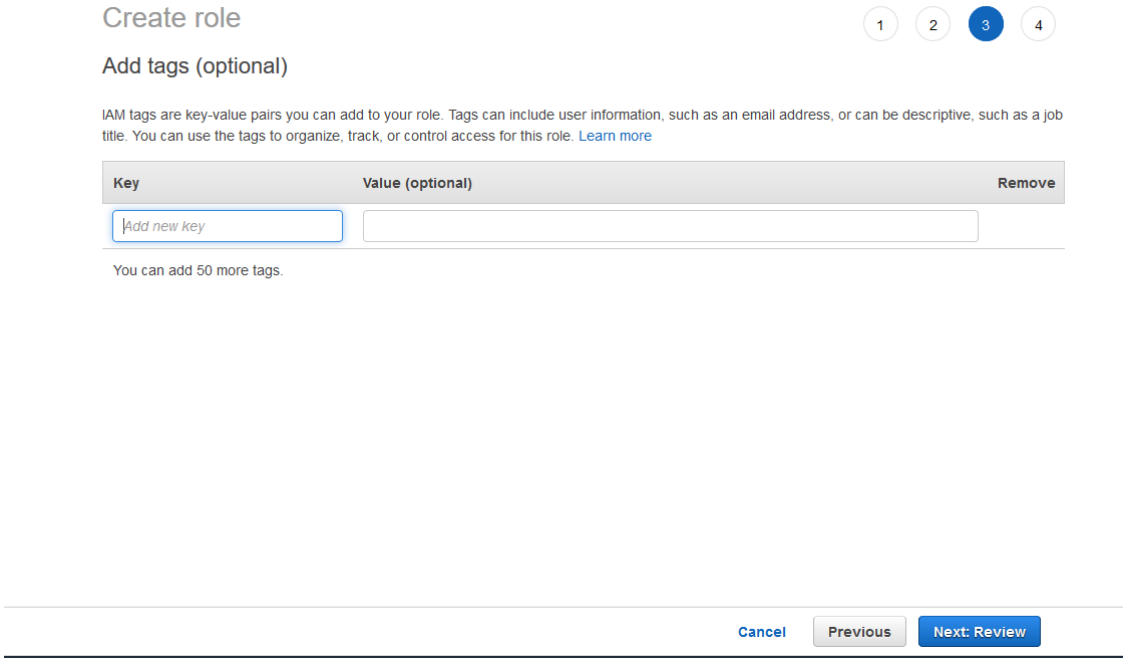


5. 在创建角色页面附加权限，将这些权限保持目前不变，然后选择下一步:s (标签。

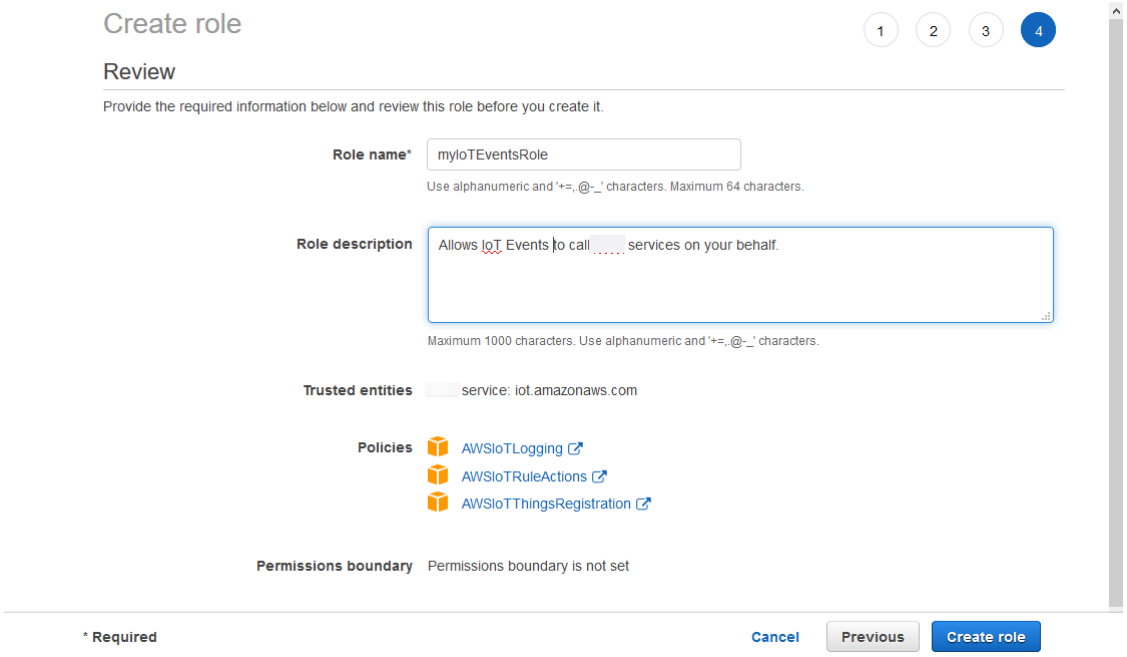
   您可在以后的步骤中添加新策略，以授予所需权限。



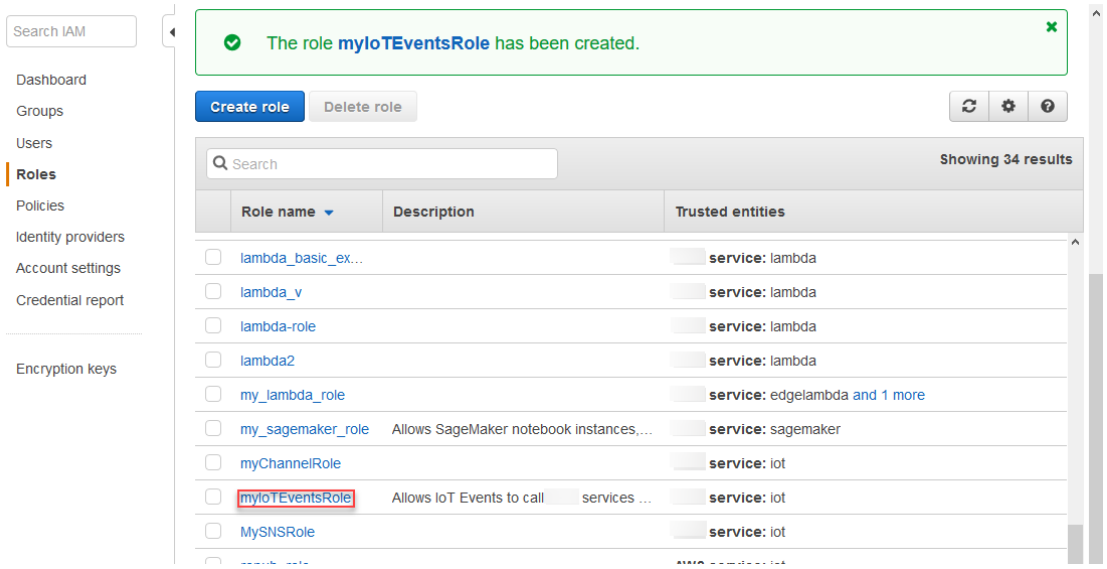
6. 在创建角色页面添加标签（可选），现在不要添加标签然后选择下一步:审核.



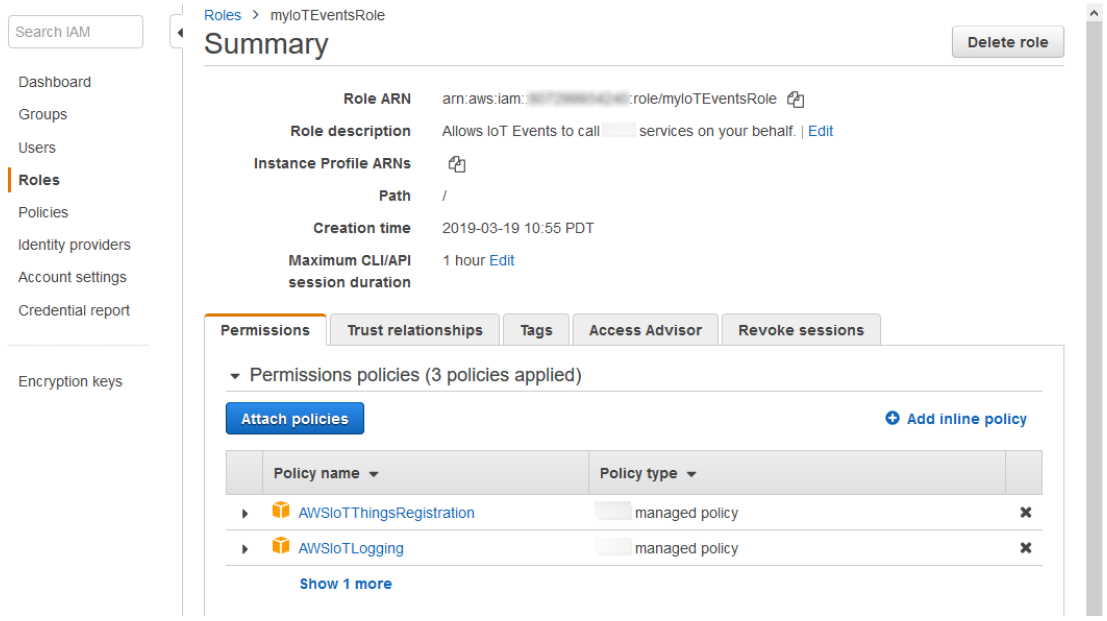
7. 在审核页面，输入角色名称，可选角色描述，然后选择创建角色.



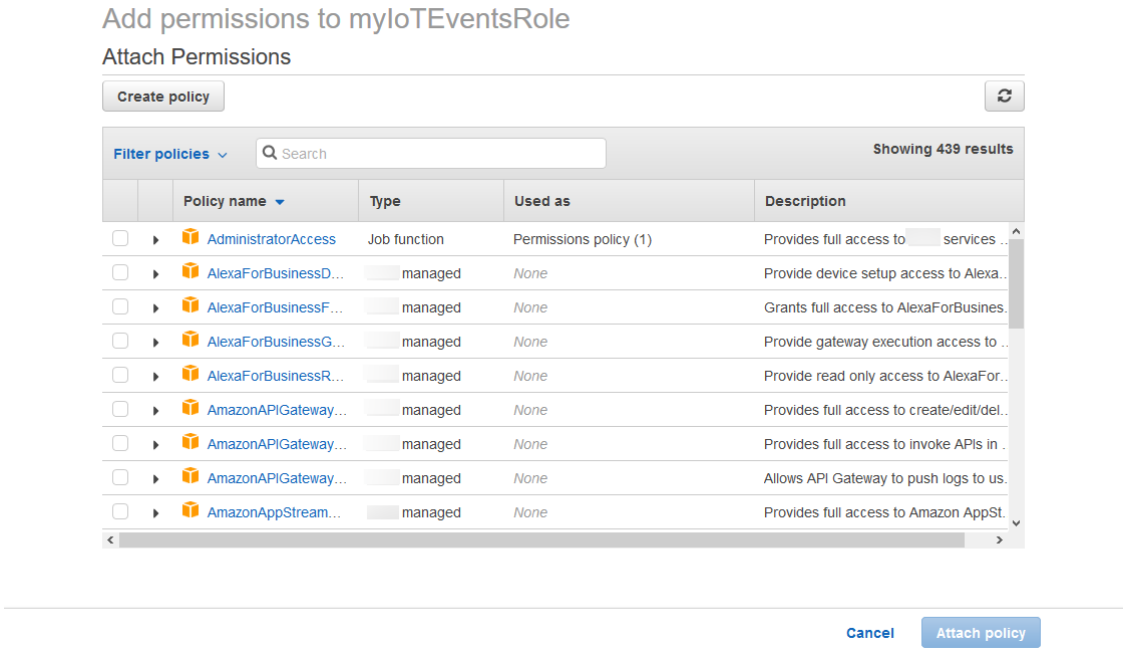
8. 在角色页面上，查找并选择您创建的角色的名称。



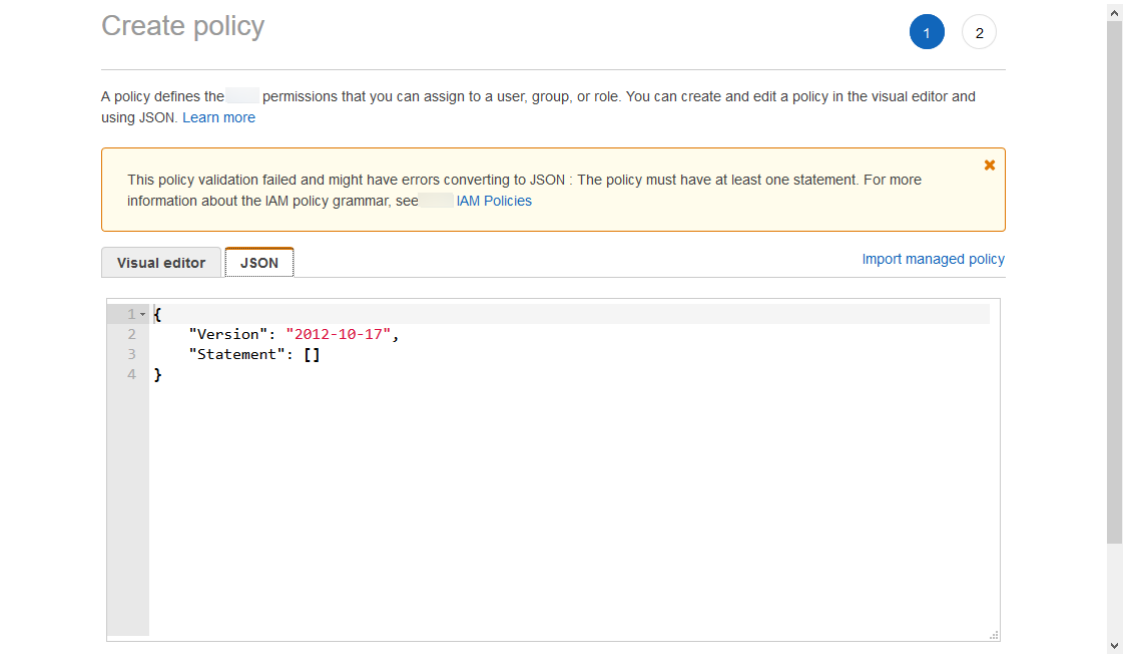
9. 在摘要你的角色页面，在Permissions (权限)选项卡，选择附加策略.



10. 在附加权限页面，选择创建策略.



11. 在创建策略页面上，选择JSON选项卡。如果出现策略验证失败警告，请选择X图标将其关闭。



12. 在JSON选项卡，请执行以下操作。

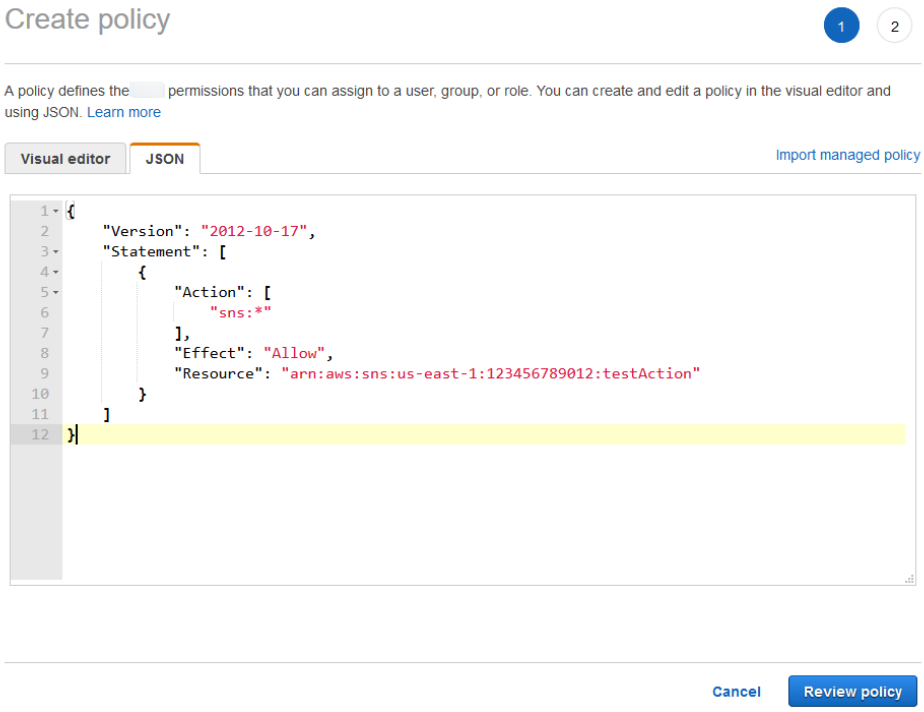
    a. 在编辑器中使用以下示例替换 JSON。

    b. 更改`Resource`值转换为 ARN。Amazon SNS 主题在中使用Amazon IoT Events.

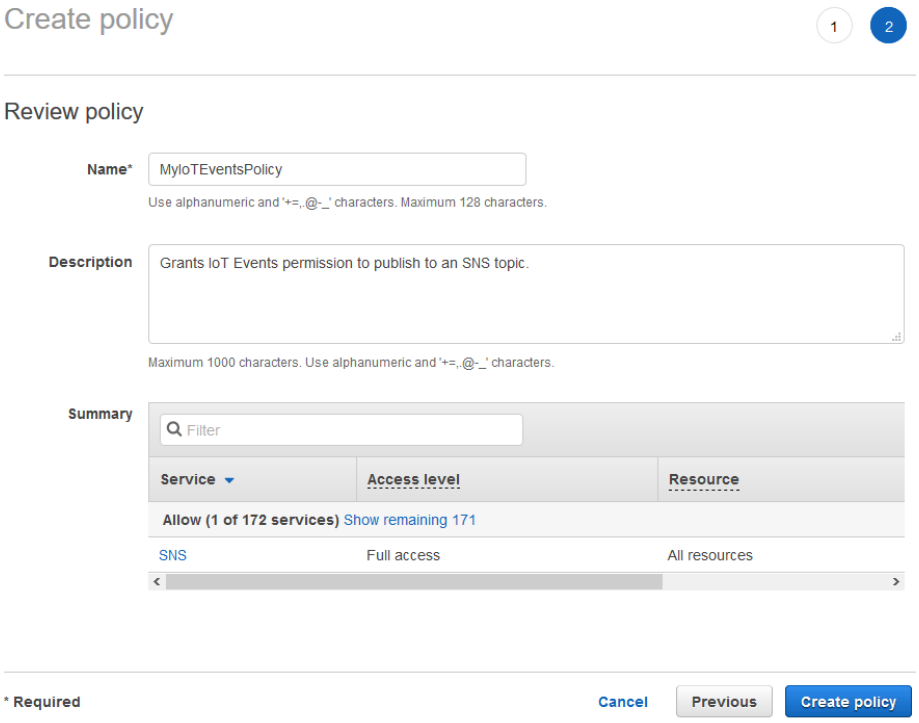
    c. 选择Review policy（查看策略）。您也可以在 Amazon SNS 主题 ARN 中使用通配符来授予更广泛的权限，但请注意由此引发的安全问题。

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

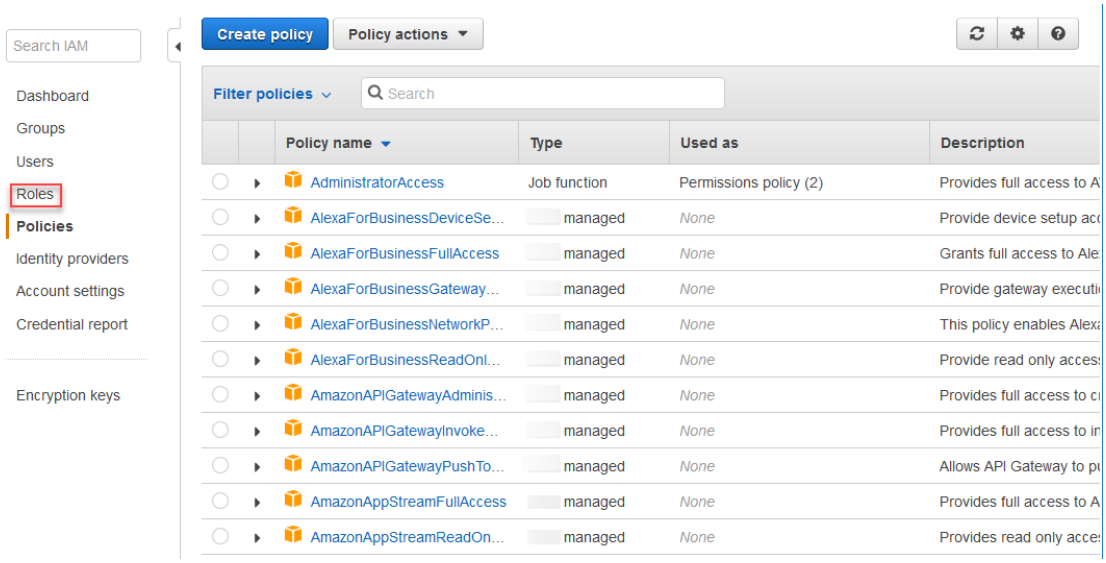
您的保单应类似以下内容。



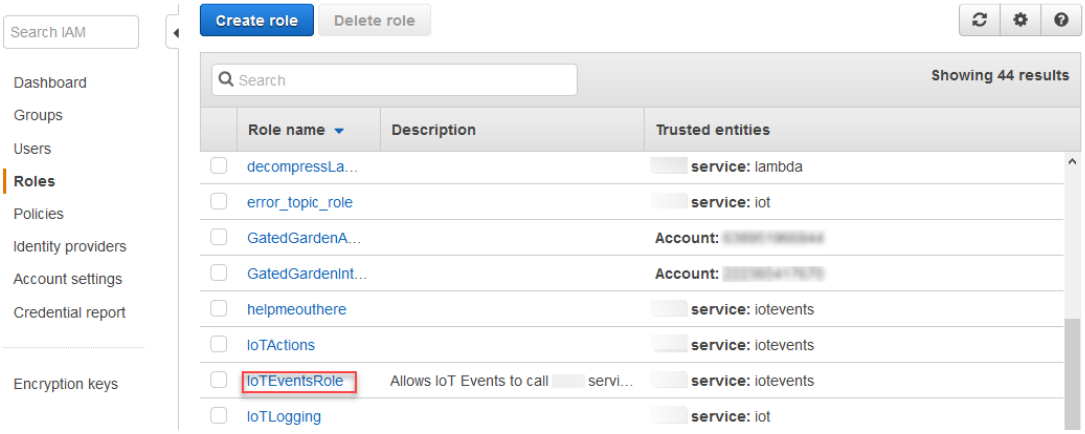
13. 在查看策略页面，输入名称对于该策略，是可选的说明，然后选择创建策略.



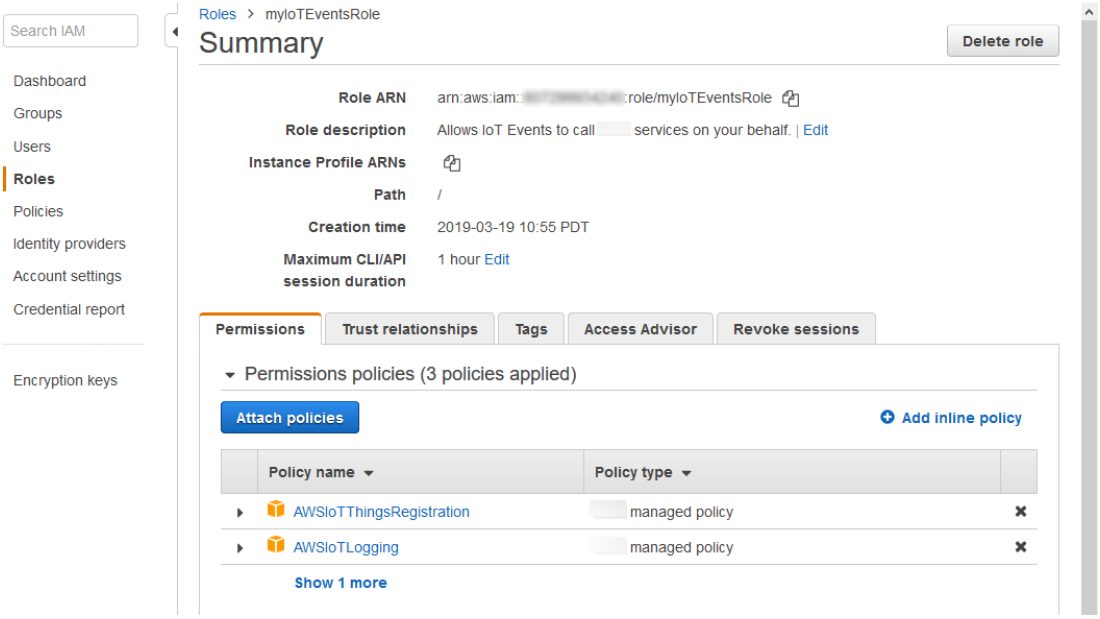
14. 在策略页面上，在导航窗格中，选择角色.



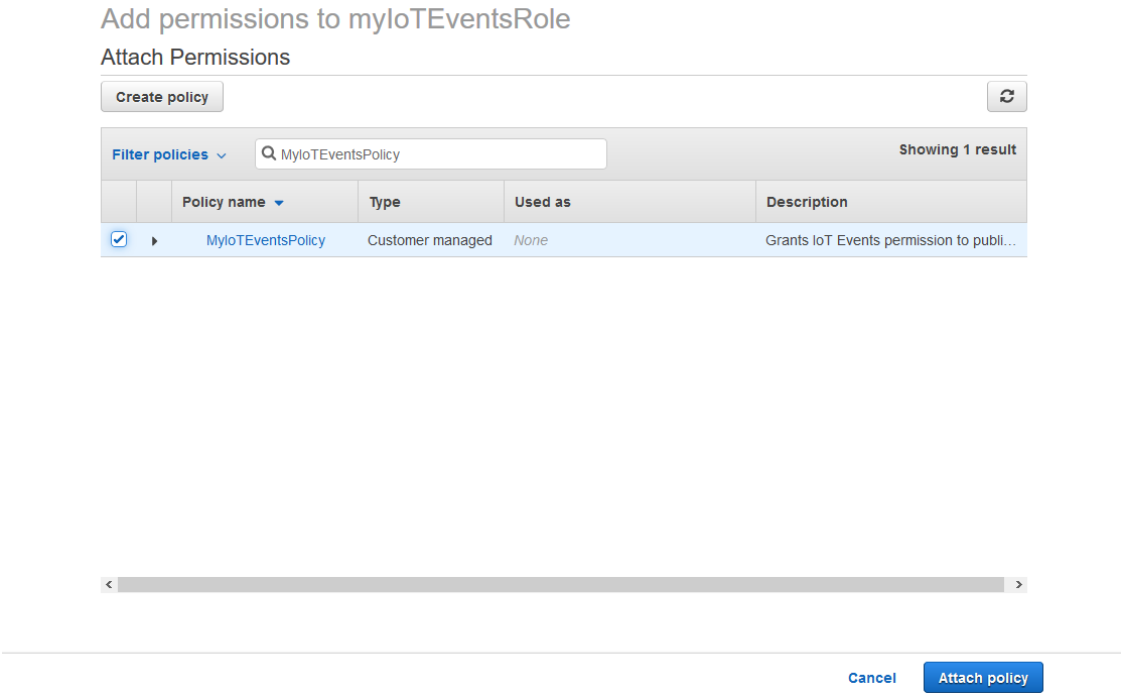
15. 在角色页面上，查找并选择您创建的角色的名称。



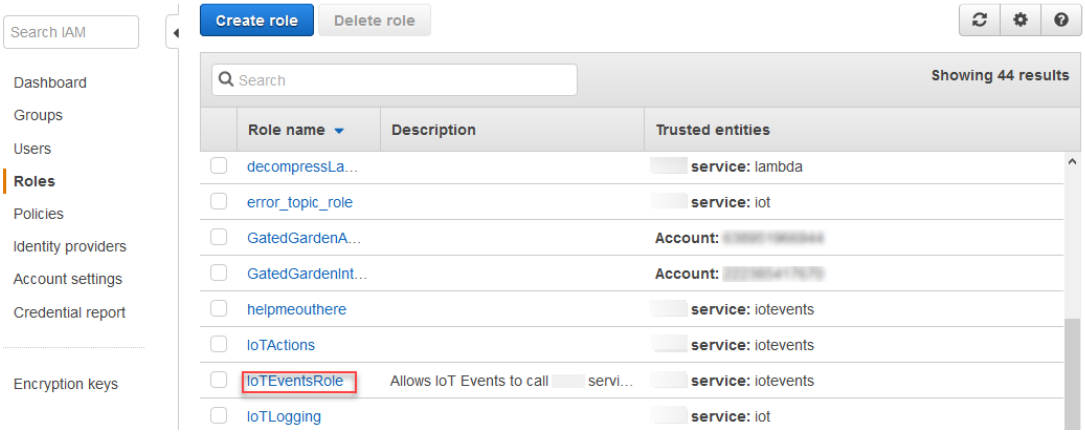
16. 在摘要你的角色页面，在Permissions (权限)选项卡，选择附加策略.



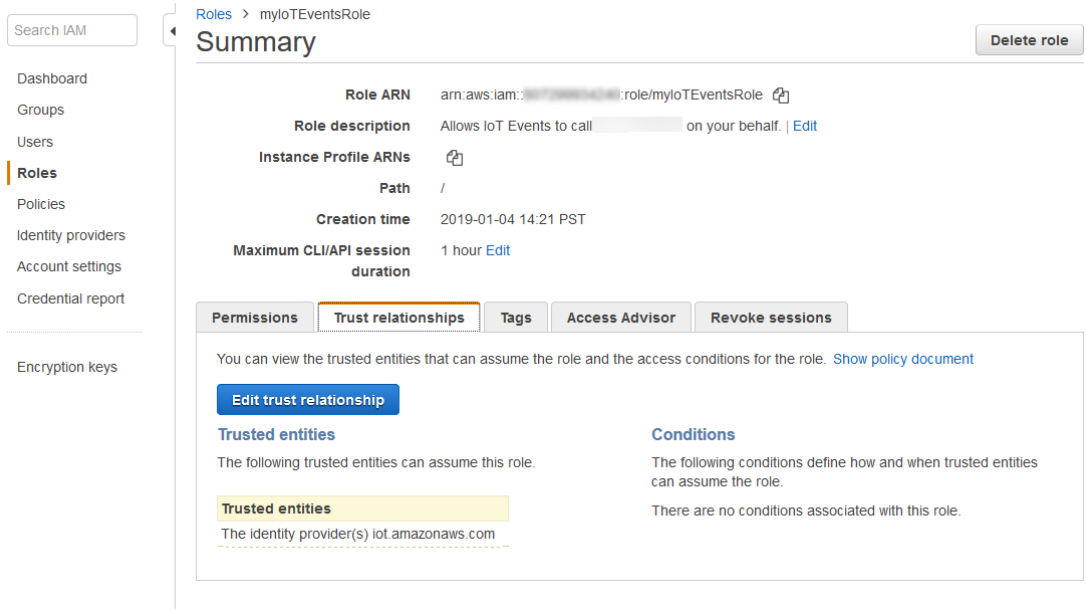
17. 在附加权限” 页面，选中您创建的策略旁边的复选框，然后选择附加策略.



18. 在角色页面上，查找并选择您创建的角色的名称。



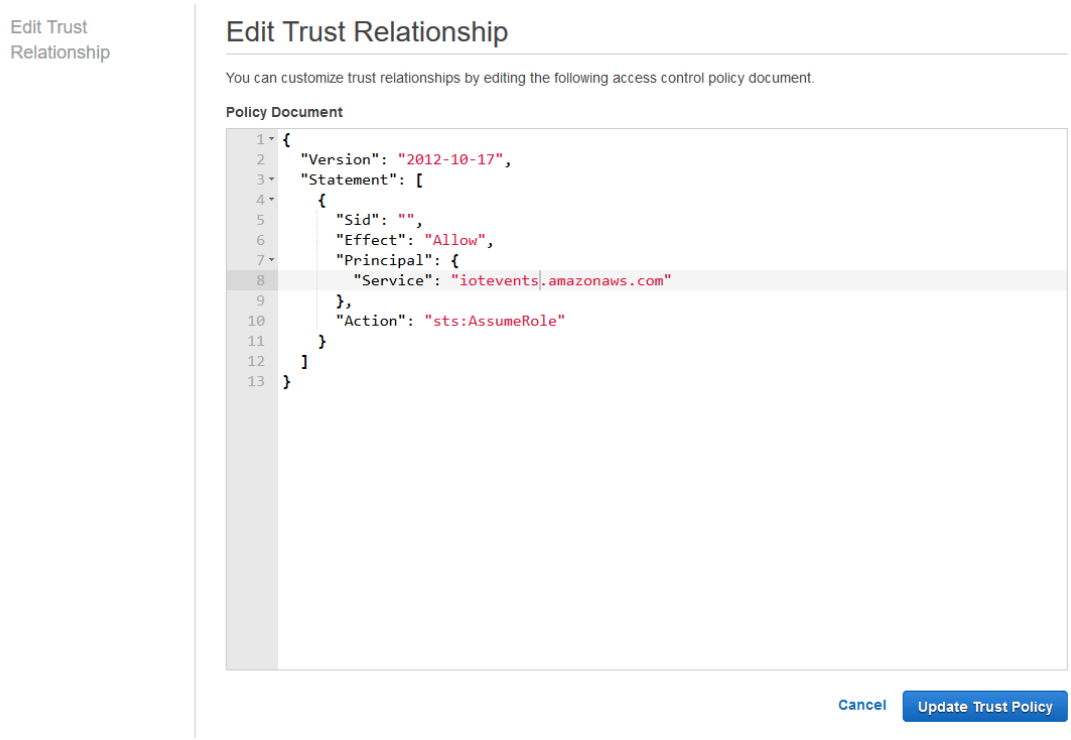
19. 在摘要该角色的页面，在信任关系选项卡，选择编辑信任关系.



20. 在编辑信任关系页面策略文档，将现有的 JSON 替换为以下内容并选择更新信任策略.

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

您的保单应类似以下内容。



你现在已经授予了Amazon IoT Events允许代表您向您的 Amazon SNS 主题发送提醒。为了增强安全性，请删除您创建的角色（默认情况下附加到您创建的角色）。

亚马逊 CloudWatch 记录角色策略

以下策略文档提供了角色策略和信任策略，可以Amazon IoT Events将日志提交到 CloudWatch 代表您们。

角色策略：

```
{
    "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:*:*:*"
            ]
        }
    ]
}
```

信任策略：

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

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

您还需要附加至 IAM 用户的 IAM permissions policy（权限策略），允许该用户传递策略，如下所示。有关更多信息，请参阅 。向用户授予权限以将角色传递给Amazon服务在里面IAM 用户指南.

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

您可使用以下命令将资源策略放入 CloudWatch 日志。这允许Amazon IoT Events将日志事件放入 CloudWatch 流流。ook

```
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\": \"*\" } ] }"
```

使用以下命令放置日志选项。替换`roleArn`使用您创建的日志角色。

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

Amazon SNS 消息角色政策

以下策略文档提供了角色策略和信任策略，可以Amazon IoT Events发送 SNS 消息。

角色策略：

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

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

信任策略：

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

开始使用Amazon IoT Events控制台

本节说明了如何使用Amazon IoT Events控制台. 您可以对发动机的两种状态进行建模：正常状态和超压状态。当发动机中测得的压力超过一定阈值时，模型会从正常状态过渡到超压状态。然后，它会发送 Amazon SNS 消息，提醒技术人员注意情况。当压力再次降至连续三个压力读数的阈值以下时，模型将返回正常状态并发送另一条 Amazon SNS 消息作为确认。

我们检查压力阈值以下的连续三个读数，以消除在非线性恢复阶段或压力读数异常的情况下可能出现的过压/正常信息卡顿现象。

在控制台上，您还可以找到几个可以自定义的预制探测器模型模板。您还可以使用控制台导入其他人编写的探测器模型，或者导出您的探测器模型并在不同的探测器中使用它们Amazon区域。如果您导入探测器模型，请确保为新区域创建所需的输入或重新创建输入，并更新所使用的所有角色 ARN。

使用Amazon IoT Events控制台以了解以下内容。

定义输入

要监控您的设备和流程，它们必须具有将遥测数据导入 Amazon IoT Events 的方法。执行此操作的方法是将消息作为输入发送到 Amazon IoT Events。有几种方式可以实现：

- 使用 BatchPutMessage操作。
- 中Amazon IoT Core写入一个Amazon IoT Events行动规则为Amazon IoT规则引擎，用于将您的消息数据转发到Amazon IoT Events. 必须按名称标识输入。
- 中Amazon IoT Analytics，使用 CreateDataset操作以创建数据集`contentDeliveryRules`. 这些规则指定了Amazon IoT Events自动发送数据集内容的输入。

在您的设备能够以这种方式发送数据之前，必须定义一个或多个输入。为此，请为每个输入指定一个名称，并指定输入将监视传入消息数据中的哪些字段。

创建探测器模型

定义一个探测器模型（您的设备或过程的模型）使用状态. 对于每种状态，定义条件（布尔值）逻辑，该逻辑评估传入的输入以检测重要事件。检测到事件后，它可以使用其他 Amazon 服务更改状态或触发自定义或预定义的操作。您可以定义其他事件，这些事件将在进入或退出某个状态以及满足某个条件（可选）时触发动作。

在本教程中，当模型进入或退出特定状态时，您将发送 Amazon SNS 消息作为操作。

监控设备或进程

如果您监视多个设备或进程，请在每个输入中指定一个字段来标识输入来自的特定设备或进程。请参阅`key`字段中/省`CreateDetectorModel`. 当识别出一台新设备时（在由该设备标识的输入字段中看到一个新值）`key`），则会创建一个探测器。每个探测器都是探测器模型的一个实例。新的探测器会继续响应来自该设备的输入，直到更新或删除其探测器模型。

如果您监控单个进程（即使有多个设备或子进程正在发送输入），则无需指定唯一标识`key`字段中返回的子位置类型。在这种情况下，会在第一个输入到达时创建单个检测器（实例）。

将消息作为输入发送到您的探测器模型

有多种方法可以将来自设备或进程的消息作为输入发送到Amazon IoT Events检测器，不需要您对消息进行其他格式化。在本教程中，您将使用Amazon IoT控制台写一个Amazon IoT Events行动规则为Amazon IoT规则引擎，用于将您的消息数据转发到Amazon IoT Events.

为此，请按名称识别输入，然后继续使用Amazon IoT控制台生成消息，这些消息作为输入转发到Amazon IoT Events.

Note

本教程使用控制台来创建。`input`和`detector model`如示例所示教程 (p. 53). 您可以使用此 JSON 示例来帮助您学习本教程。

主题

先决条件

如果您没有Amazon账户，创建一个。

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。
3. 创建两个Amazon Simple Notification Service (Amazon SNS) 主题。

   本教程（和相应的示例）假设您创建了两个 Amazon SNS 主题。这些主题的 ARN 显示为：`arn:aws:sns:us-east-1:123456789012:underPressureAction`和`arn:aws:sns:us-east-1:123456789012:pressureClearedAction`. 用您创建的 Amazon SNS 主题的 ARN 替换这些值。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南。

   除了向 Amazon SNS 主题发布警报外，您还可以让探测器发送带有您指定主题的 MQTT 消息。使用此选项，您可以使用以下方法验证您的探测器模型是否正在创建实例，以及这些实例是否正在发送警报Amazon IoT核心控制台用于订阅和监控发送到这些 MQTT 主题的消息。您还可以在运行时使用在探测器模型中创建的输入或变量动态定义 MQTT 主题名称。
4. 选择一个Amazon支持的地区Amazon IoT Events. 有关更多信息，请参阅 。Amazon IoT Events在Amazon一般参考. 有关帮助信息，请参阅使用Amazon Web Services Management Console在开始使用Amazon Web Services Management Console.

创建输入

当您为模型构建输入时，我们建议收集包含示例消息负载的文件，您的设备或进程将发送这些文件来报告其运行状况。拥有这些文件将帮助您定义所需的输入。

您可以通过本节中描述的多种方法创建输入。

要开始使用，请创建一个名为的文件`input.json`在您的本地文件系统中，包含以下内容：

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

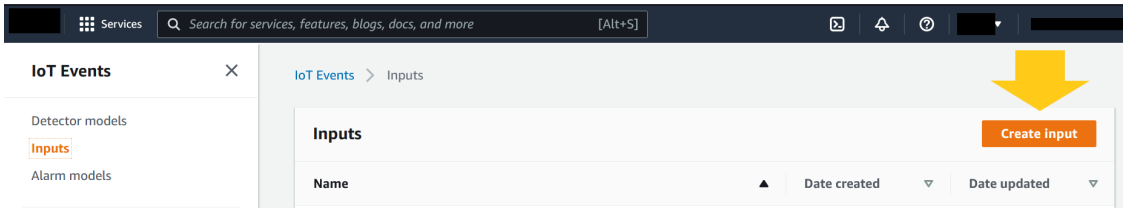
现在你有这个初学者了`input.json`文件，你可以创建一个输入。使用本节中的主题之一获取有关使用导航窗格或使用探测器模型创建输入的说明。

主题

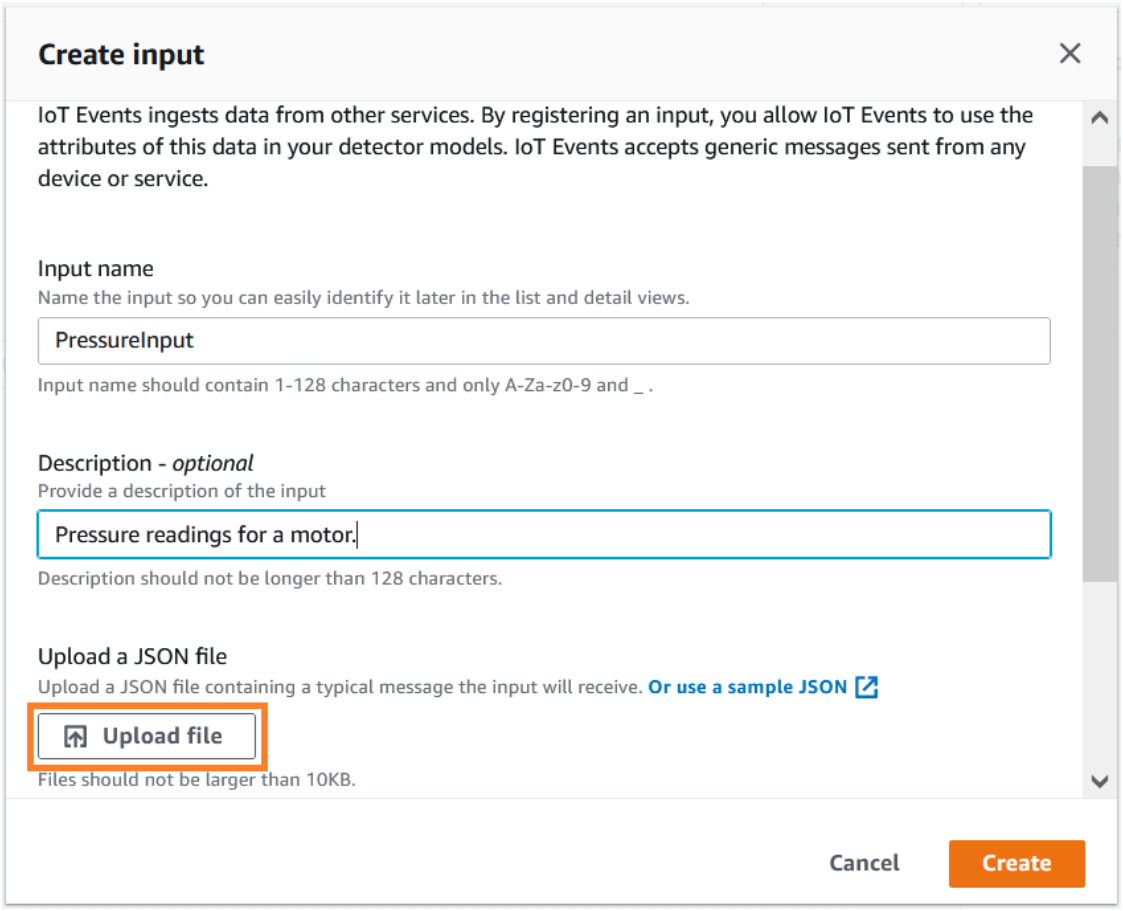
在导航窗格中创建输入

本主题演示如何创建输入，对于警报模型或探测器模型，请通过导航窗格。

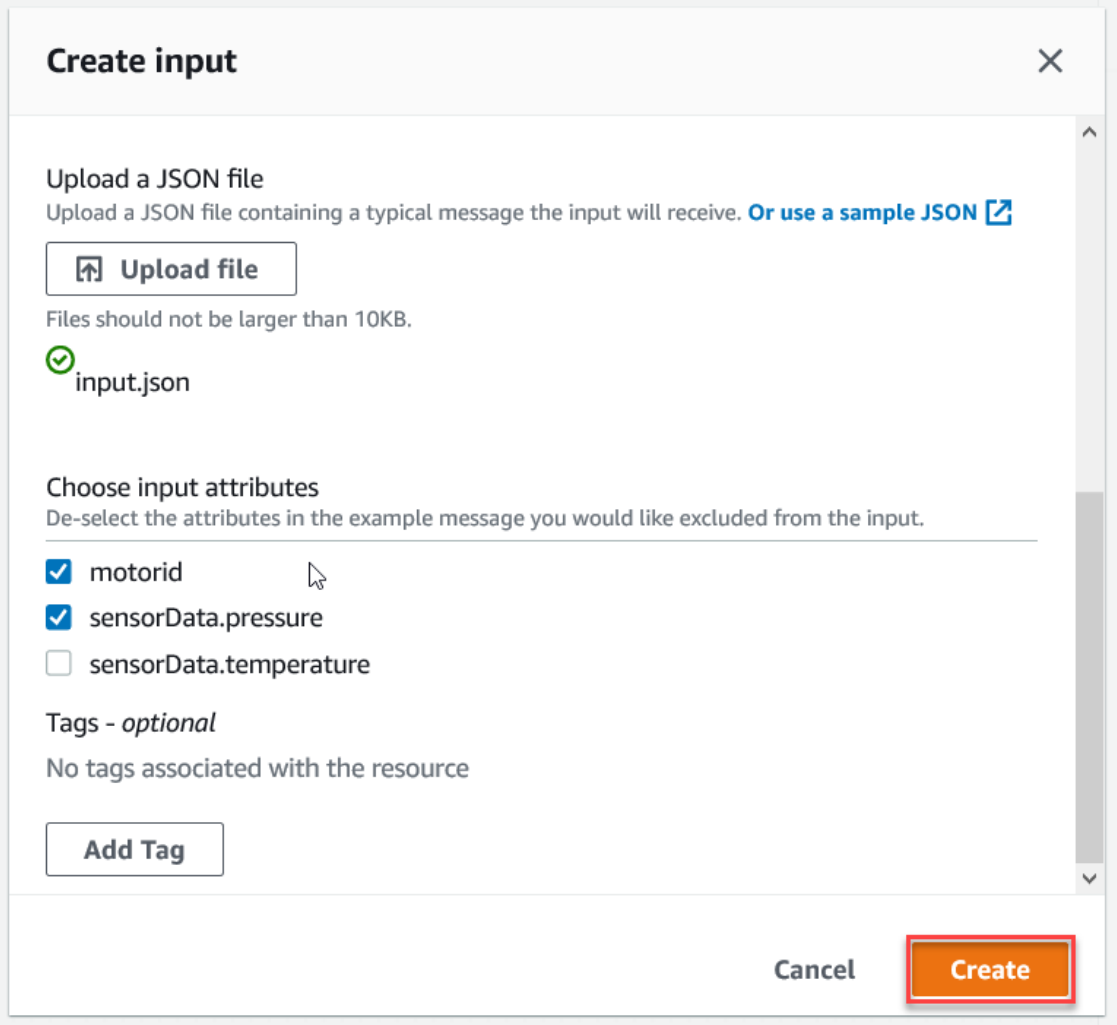
1. 登录Amazon IoT Events控制台或者选择 “新建” 选项Amazon IoT Events账户。
2. 在Amazon IoT Events控制台，在左上角，选择并展开导航窗格。
3. 在左侧导航窗格中，选择输入.
4. 在控制台的右上角，选择创建输入.



5. 对于输入，输入InputName，可选说明，然后选择上传文件. 在显示的对话框中，选择`input.json`您在概述中创建的文件创建输入.



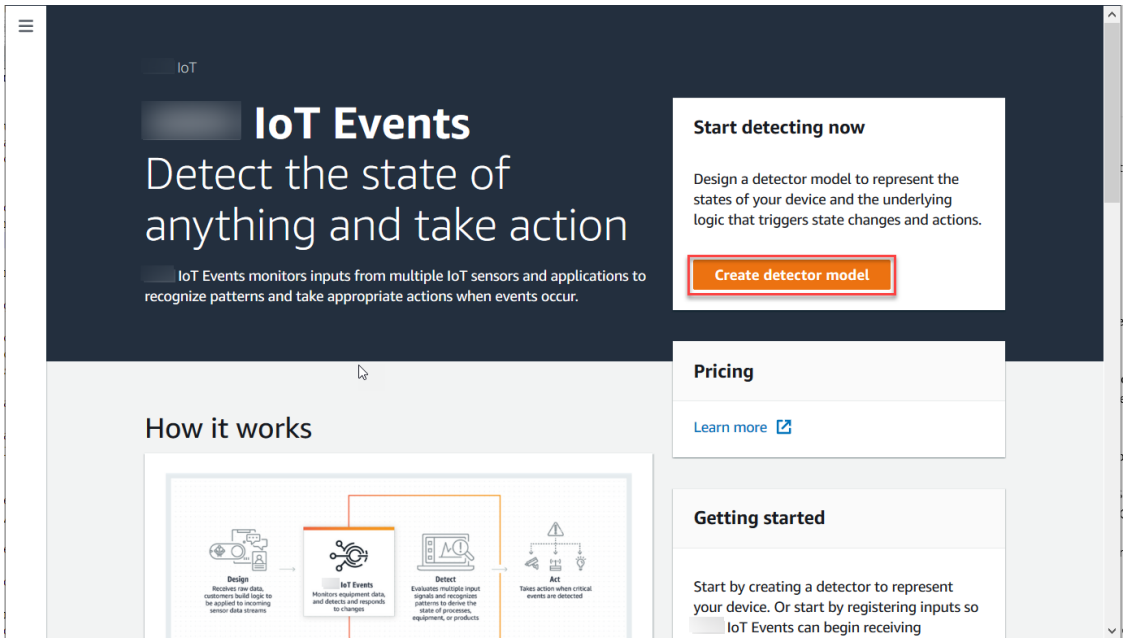
6. 对于选择输入属性，选择要使用的属性，然后选择Create. 在此示例中，我们选择motorid和传感器数据。压力.



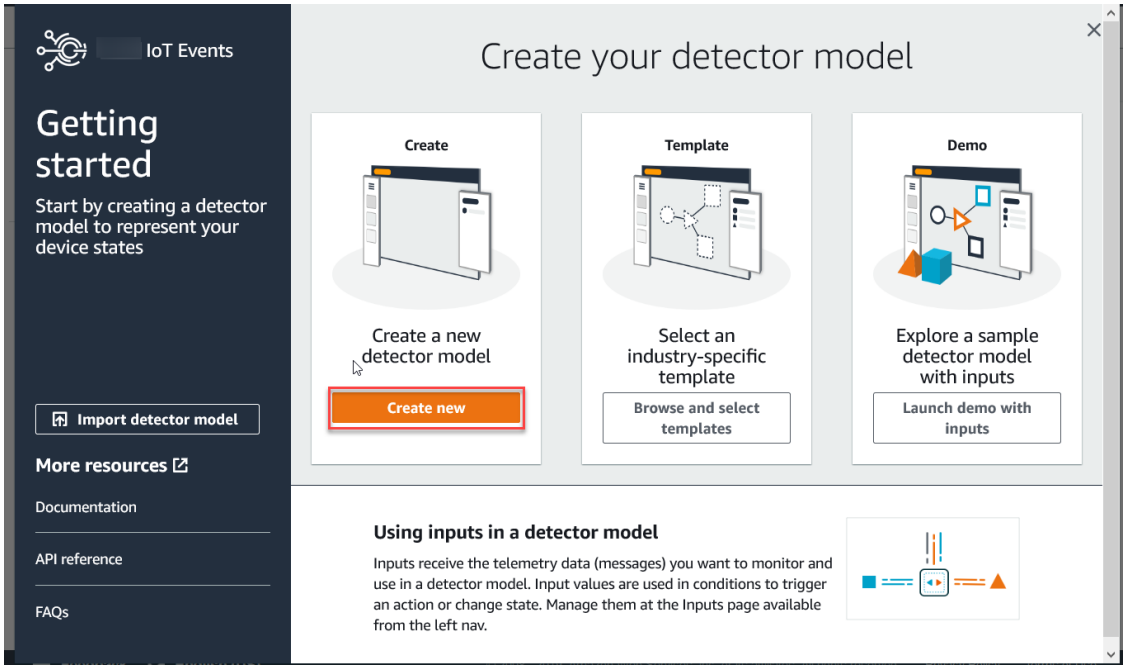
在探测器模型中创建输入

本主题说明如何定义输入让探测器模型接收遥测数据或消息。

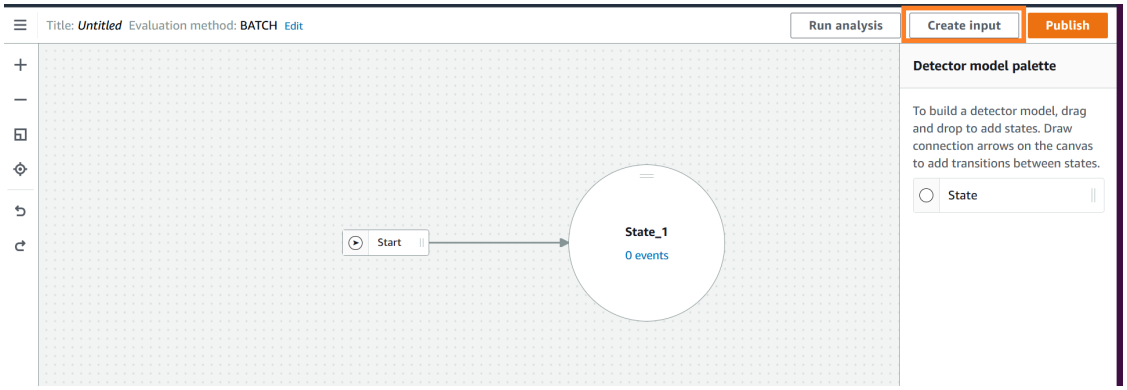
1. 打开 Amazon IoT Events 控制台。
2. 在里面Amazon IoT Events控制台，选择创建检测器模型.



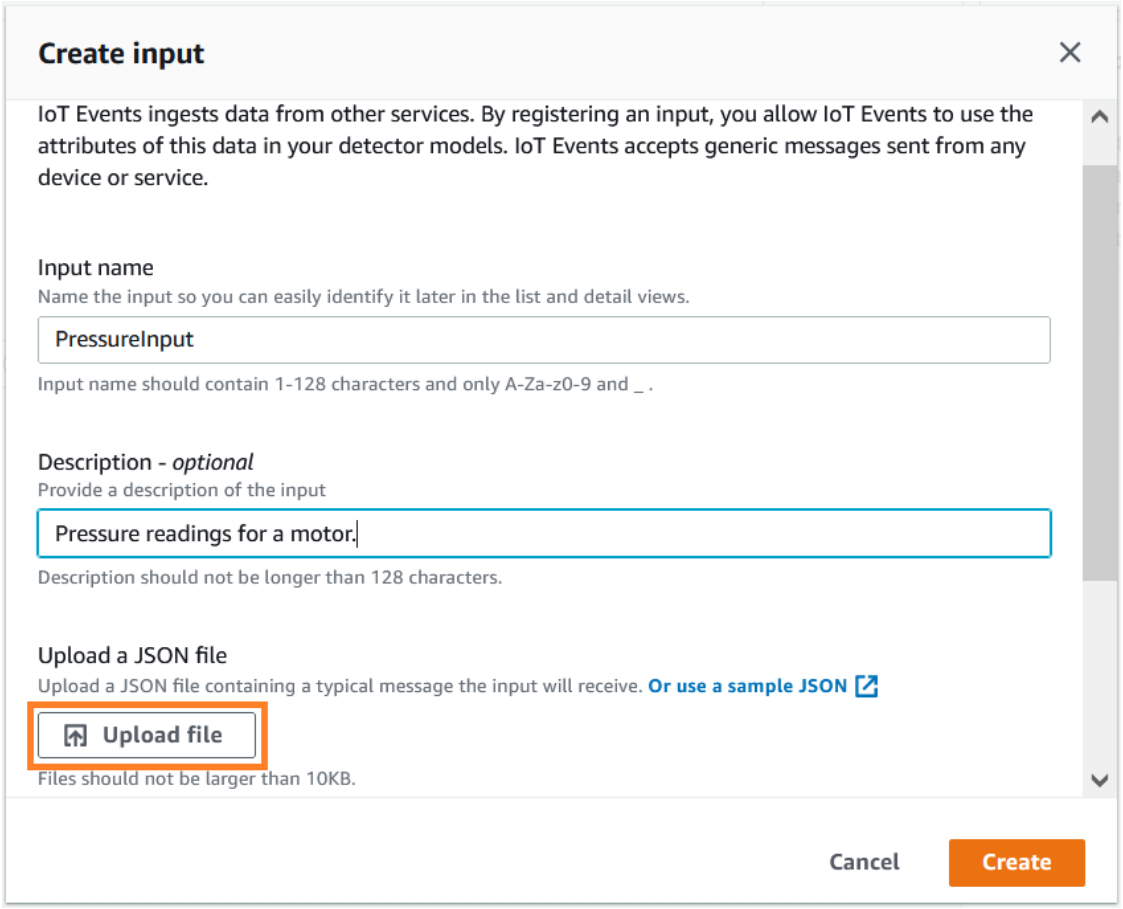
3. 选择 Create new（新建）。



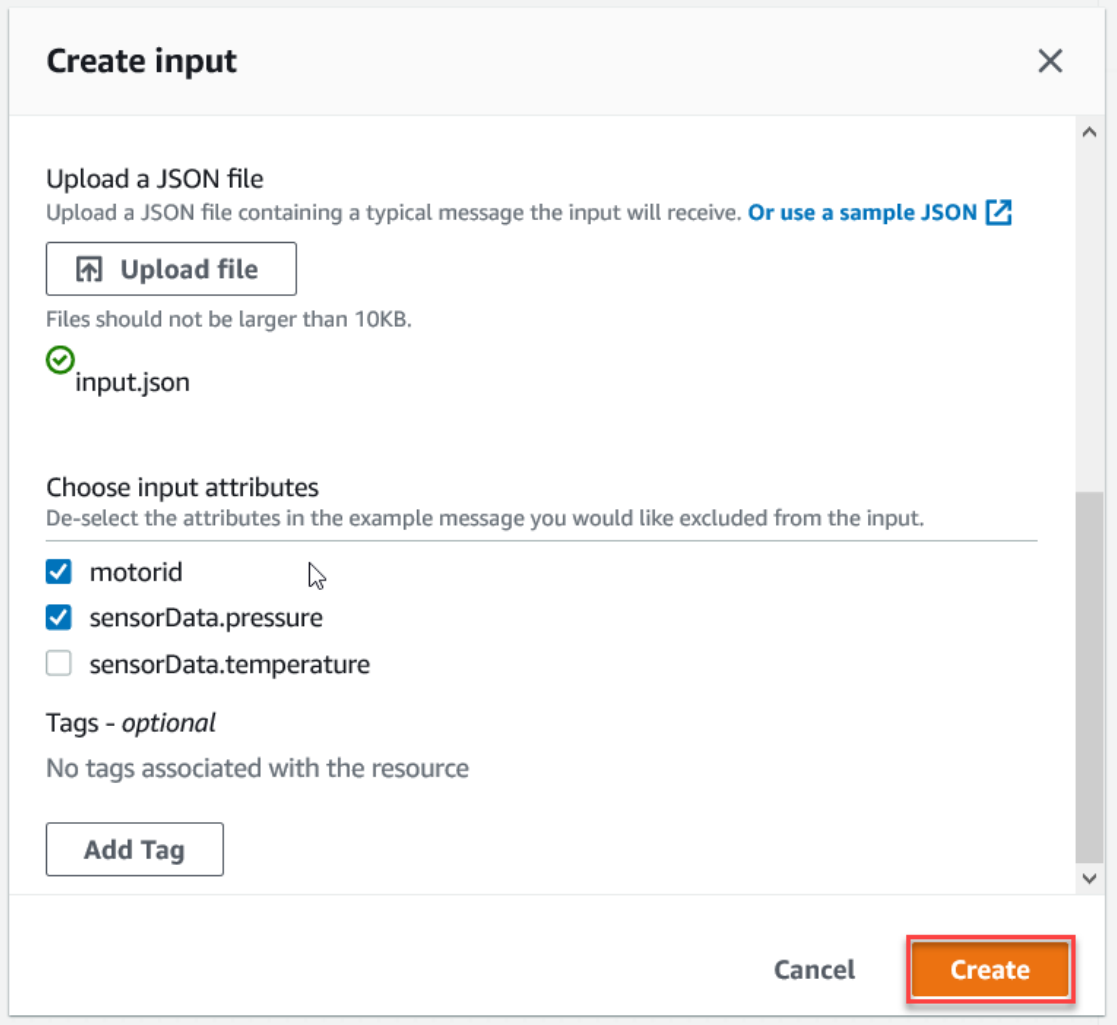
4. 选择创建输入。



5. 对于输入，输入InputName，可选说明，然后选择上传文件. 在显示的对话框中，选择`input.json`您在概述中创建的文件创建输入.



6. 对于选择输入属性，选择要使用的属性，然后选择Create. 在此示例中，我们选择motorid和sensorData.p.



创建探测器模型

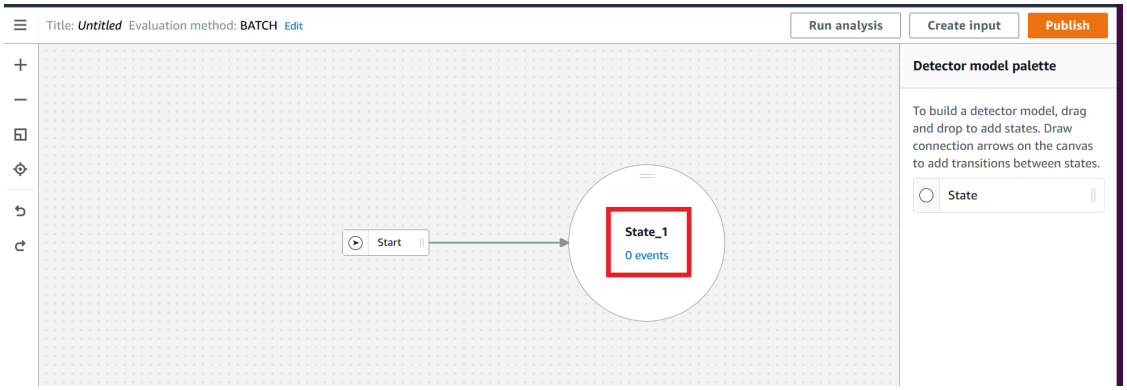
在本主题中，您将定义一个探测器模型（您的设备或过程的模型）使用状态.

对于每种状态，您定义条件（布尔值）逻辑，该逻辑评估传入的输入以检测重要事件。当检测到事件时，它会改变状态并可能触发其他操作。这些事件被称为过渡事件。

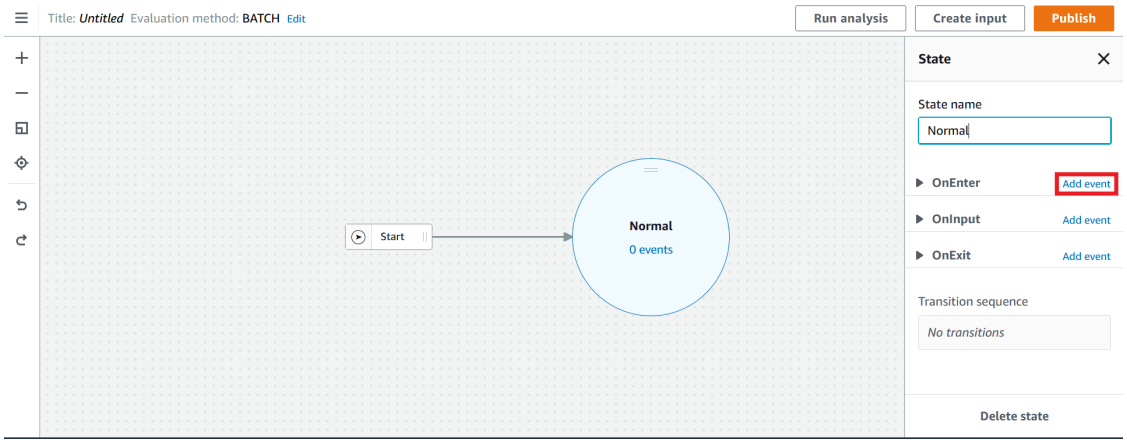
在你的状态中，你还可以定义事件，这些事件可以在探测器进入或退出该状态或接收到输入时执行动作（这些被称为`OnEnter`,`OnExit`和`OnInput`活动）。仅当事件的条件逻辑计算结果为时，才会执行操作`true`.

创建探测器模型

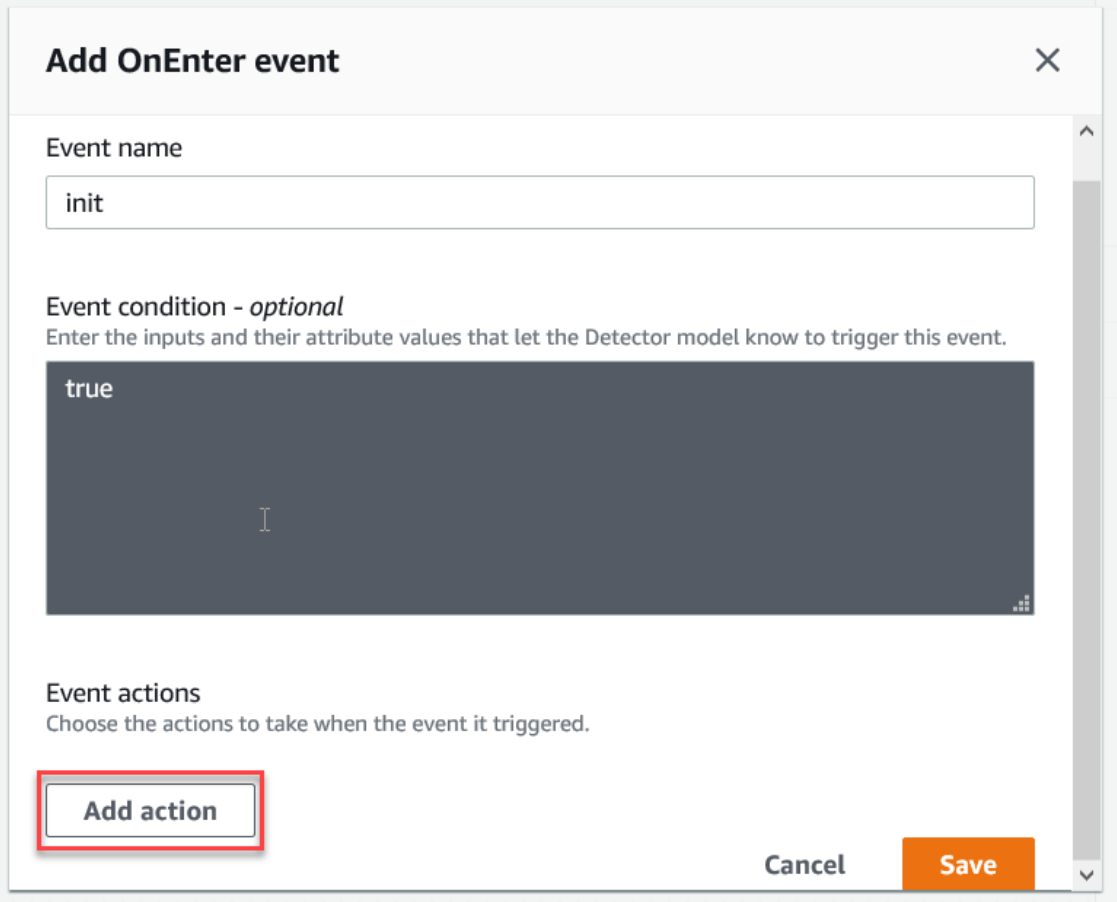
1. 已为您创建了第一个探测器状态。要对其进行修改，请选择带有标签的圆圈State_1在主编辑空间中。



2. 在州窗格中，输入州/省和OnEnter，选择添加活动.



3. 在Add OnEnter 事件页面，输入Event name (事件名称)还有事件状况. 在此示例中，输入`true`表示事件总是在进入状态时触发。
4. 下面事件件操作数，选择添加操作.



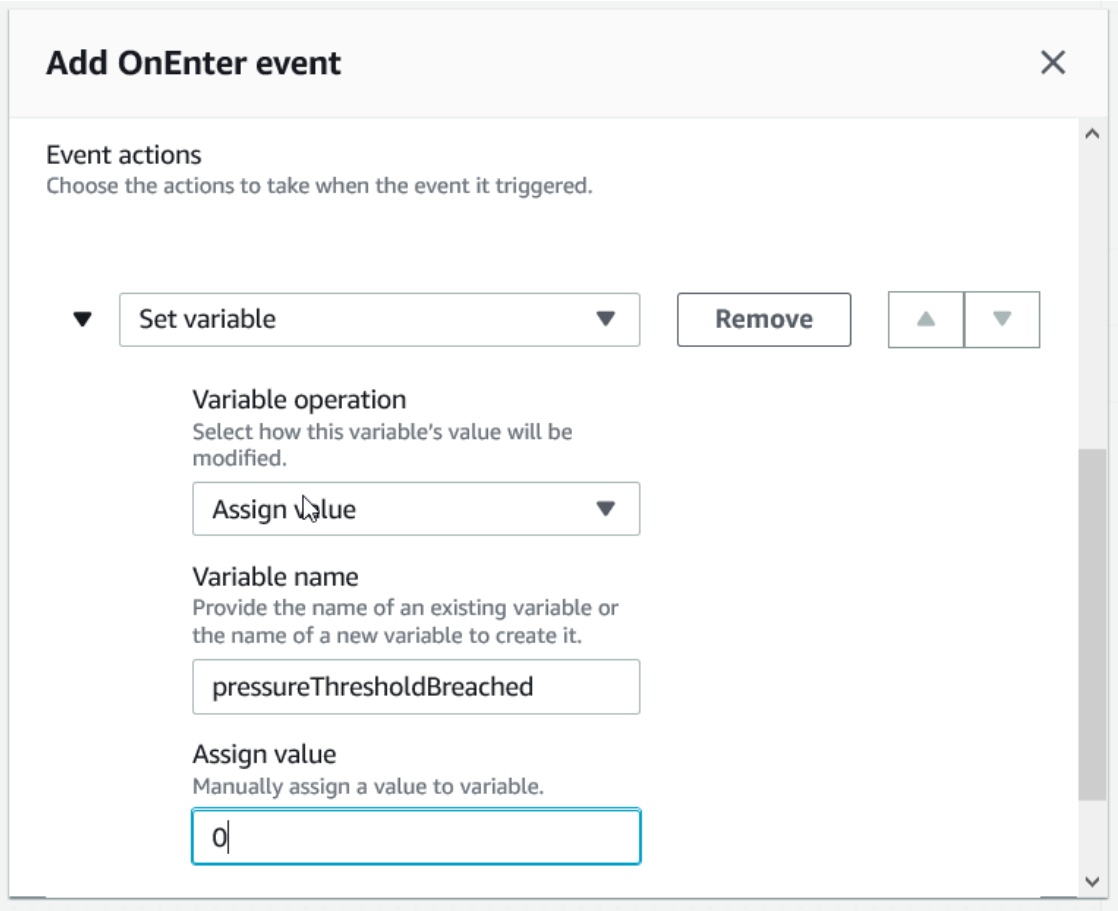
5. 下面事件件操作数，请执行以下操作：

   a. Select设置变量

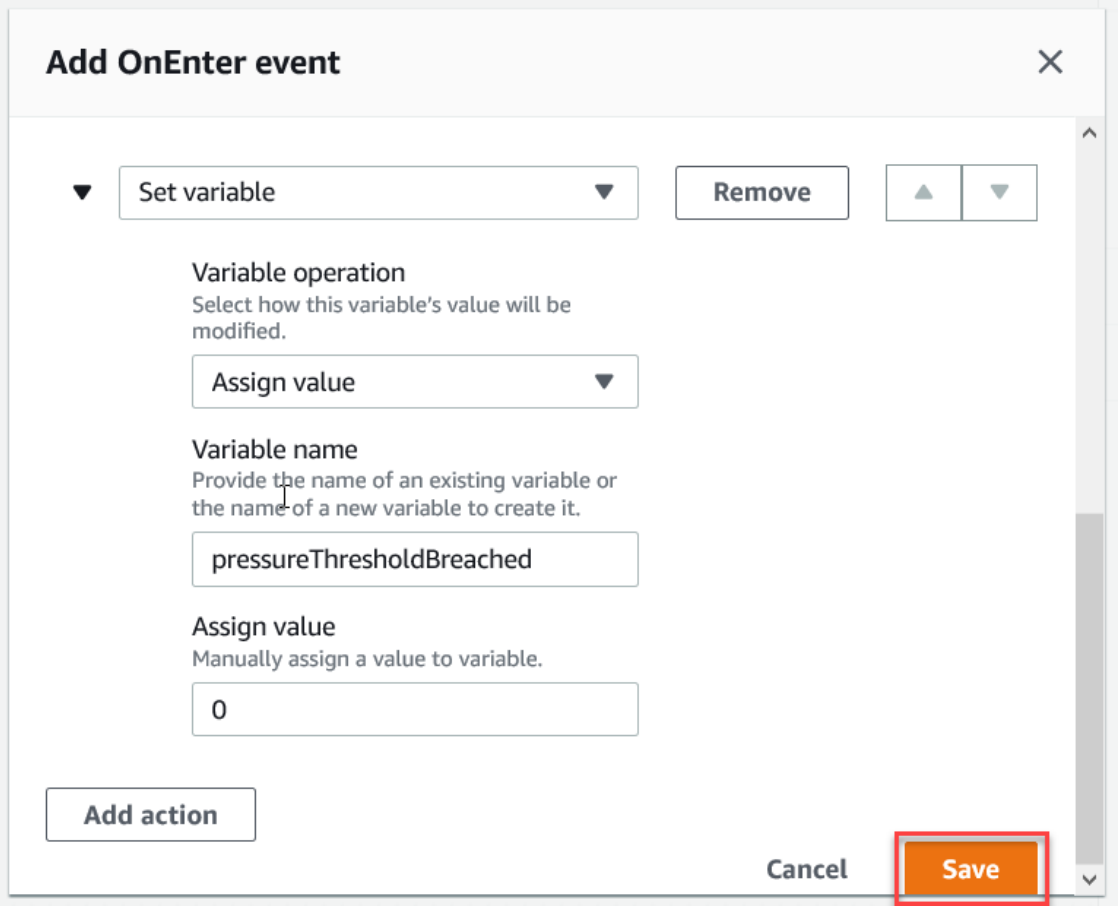
   b. 对于变量操作，选择分配价值.

   c. 对于变量名称，输入要设置的变量的名称。

   d. 对于变量值，输入值**0**（零）。

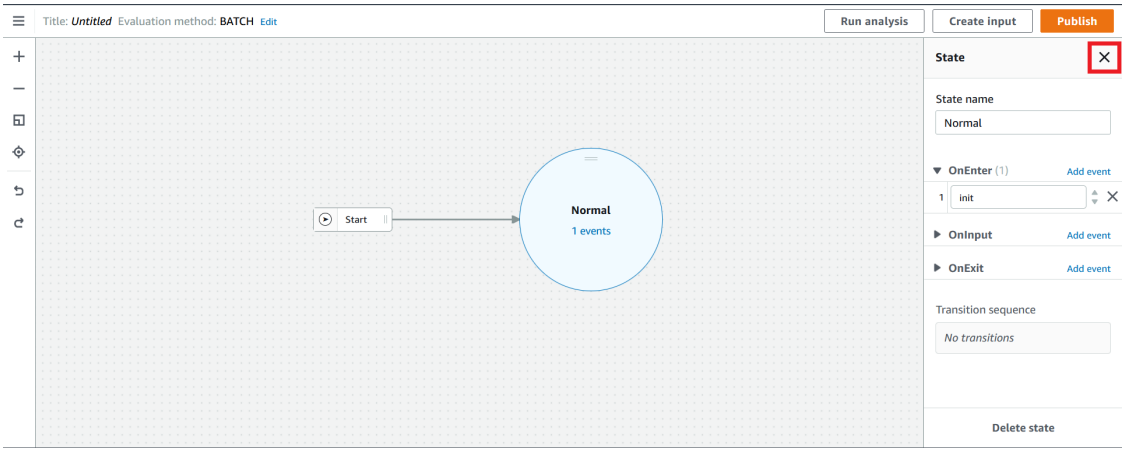

Add OnEnter event
Event actions
Choose the actions to take when the event it triggered.
Set variable
Remove
Variable operation
Select how this variable's value will be modified.
Assign value
Variable name
Provide the name of an existing variable or the name of a new variable to create it.
pressureThresholdBreached
Assign value
Manually assign a value to variable.
0

6. 选择Save（保存）。

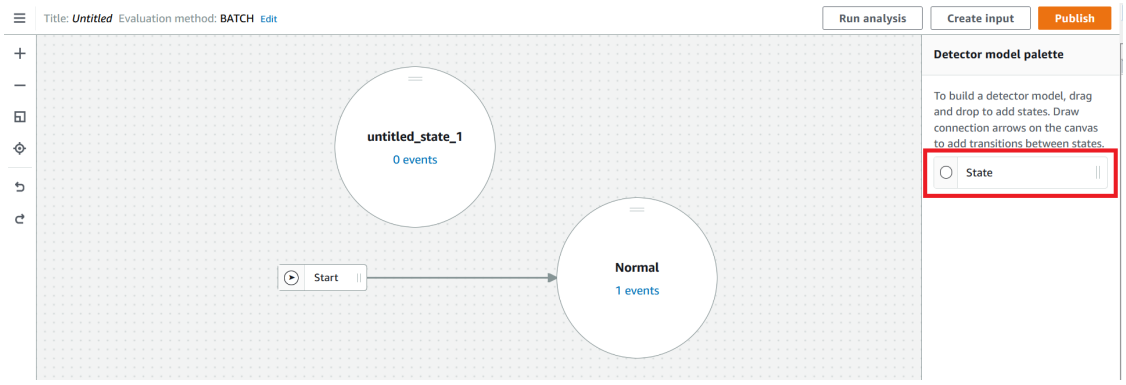


可以在探测器模型中的任何事件中设置变量（给定一个值），就像您定义的变量一样。但是只有在检测器达到状态并在定义或设置的情况下执行操作后，才能引用其值（例如，在事件的条件逻辑中）。

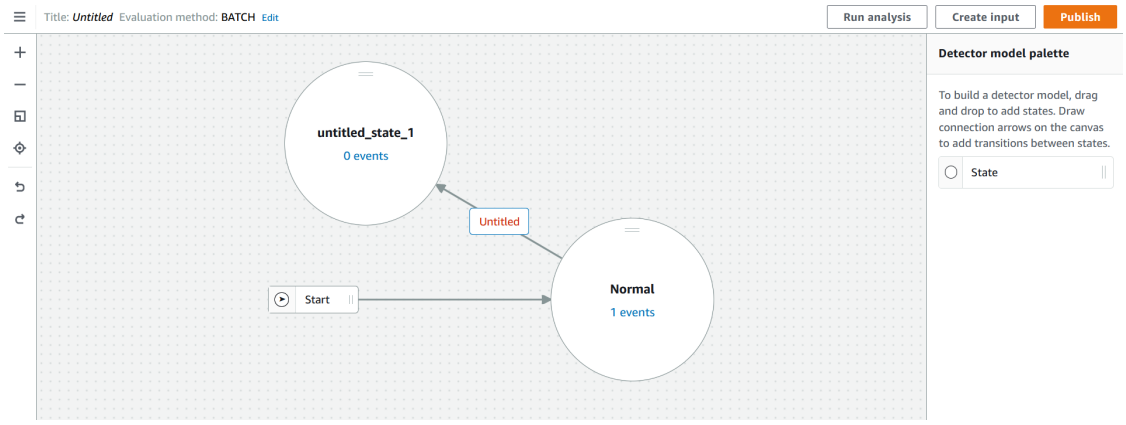
7. 在州窗格中，选择X旁边州返回到探测器模型调色板.



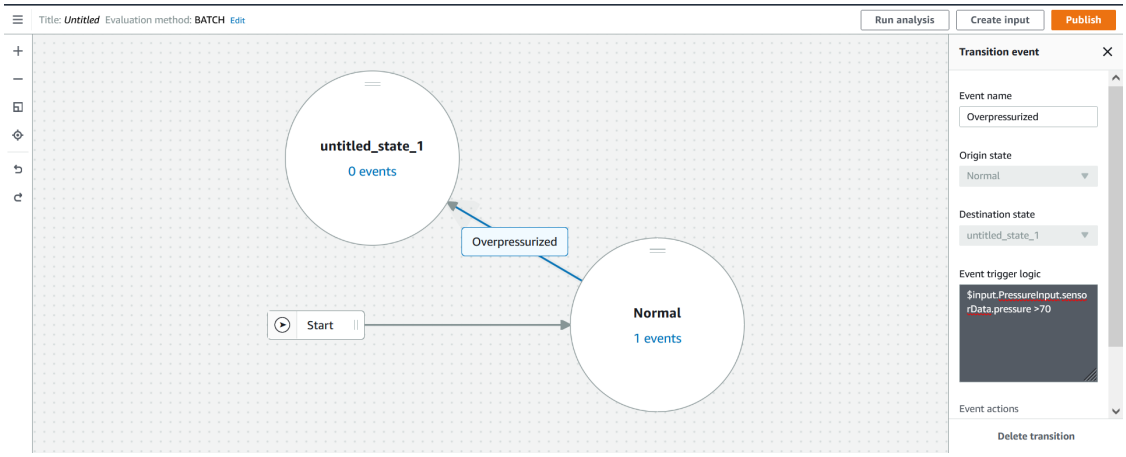
8. 要创建第二个探测器状态，请在探测器模型调色板，选择州然后将其拖到主编辑空间中。这会创建一个标题为`untitled_state_1`.



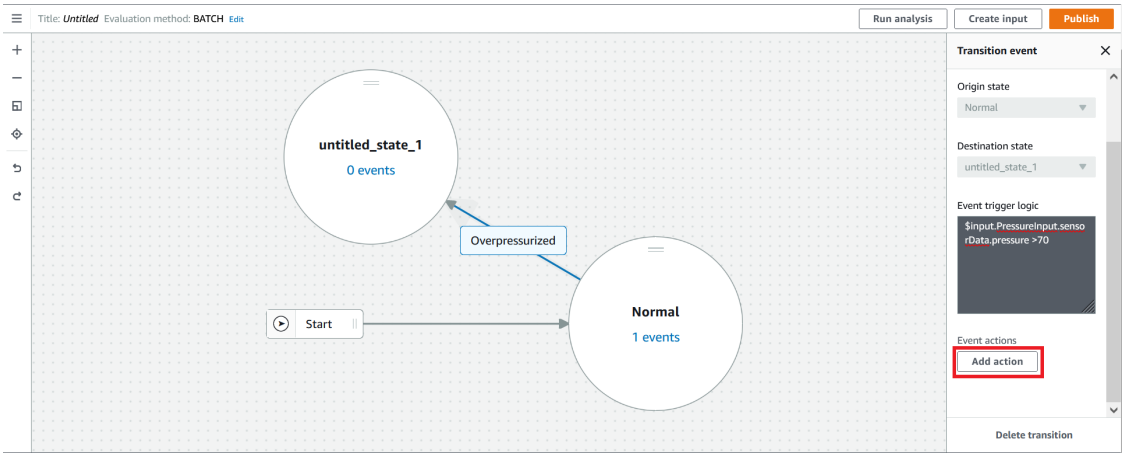
9. 在第一个状态下暂停 (普通)。州周上会出现一个箭头。
10. 单击箭头并将其从第一个状态拖动到第二个状态。从第一状态到第二状态的定向线（标记无标题）。



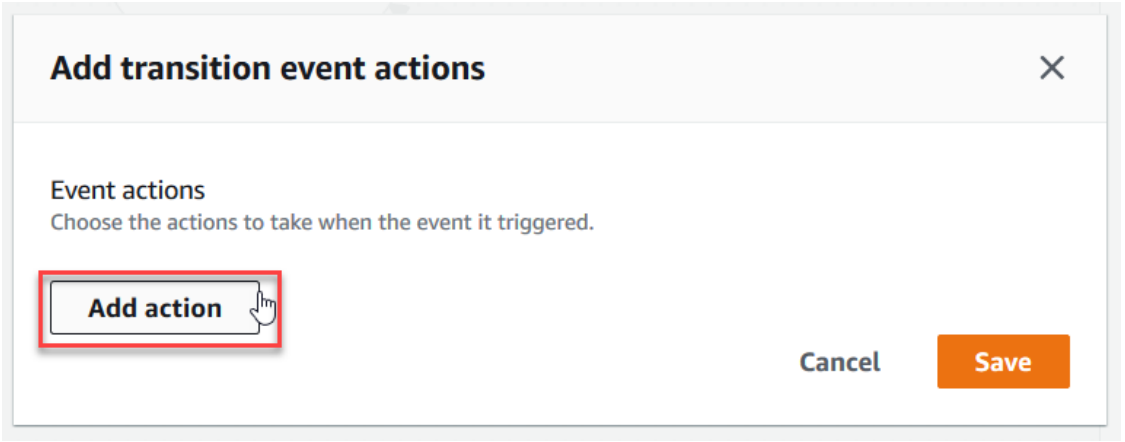
11. 选择无标题行。在过渡活动窗格，输入Event name (事件名称)和事件触发逻辑.



12. 在过渡活动窗格，选择添加操作.

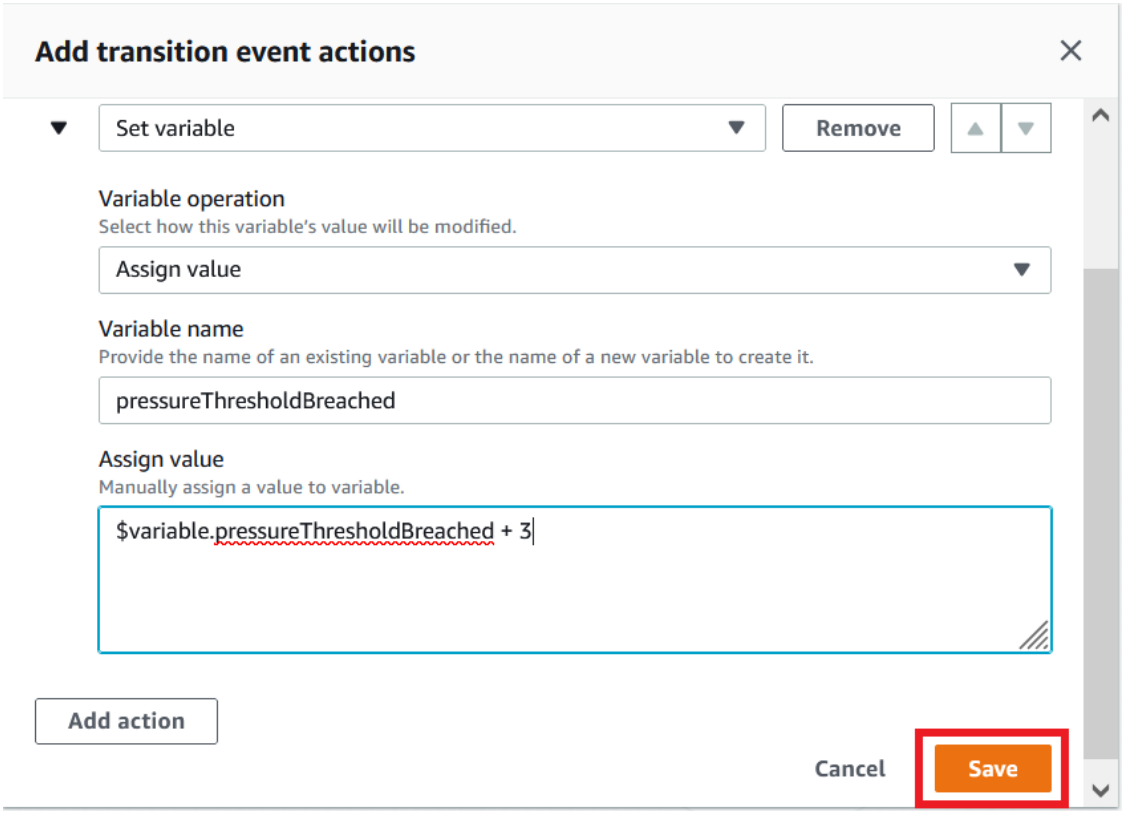


13. 在添加过渡事件动作窗格，选择添加操作.

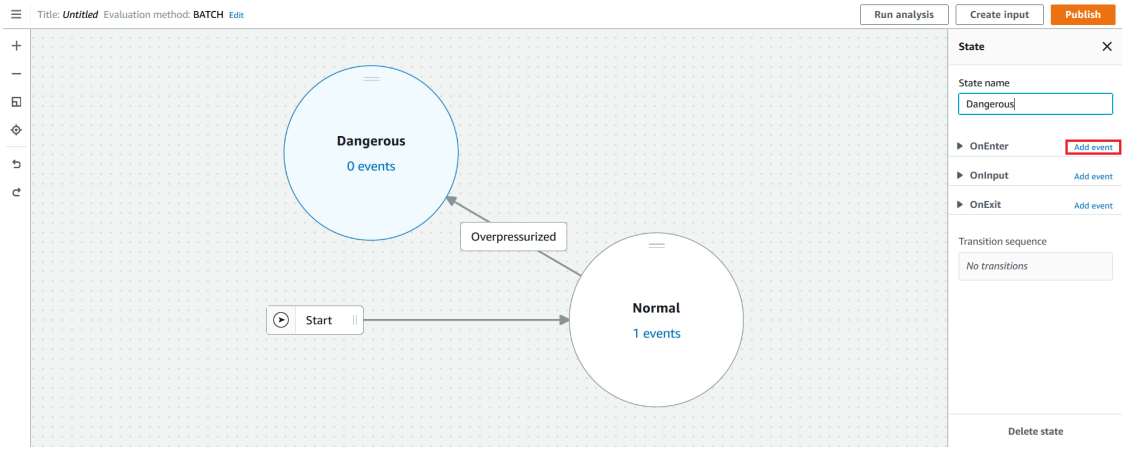


14. 对于选择一个动作，选择设置变量.

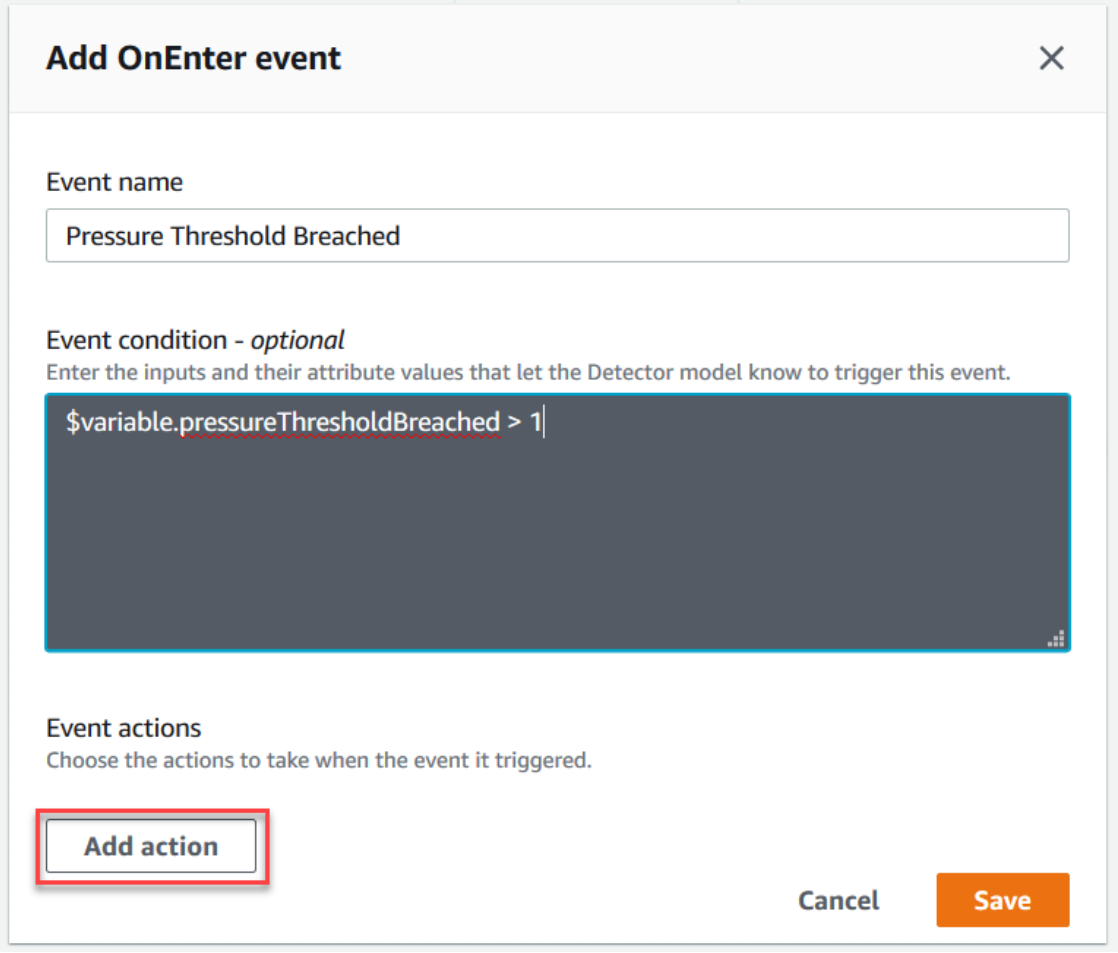
    a. 对于变量操作，选择分配价值.
    b. 对于变量名称，输入变量的名称。
    c. 对于分配价值，输入值，例如：`$variable.pressureThresholdBreached + 3`
    d. 选择Save（保存）。



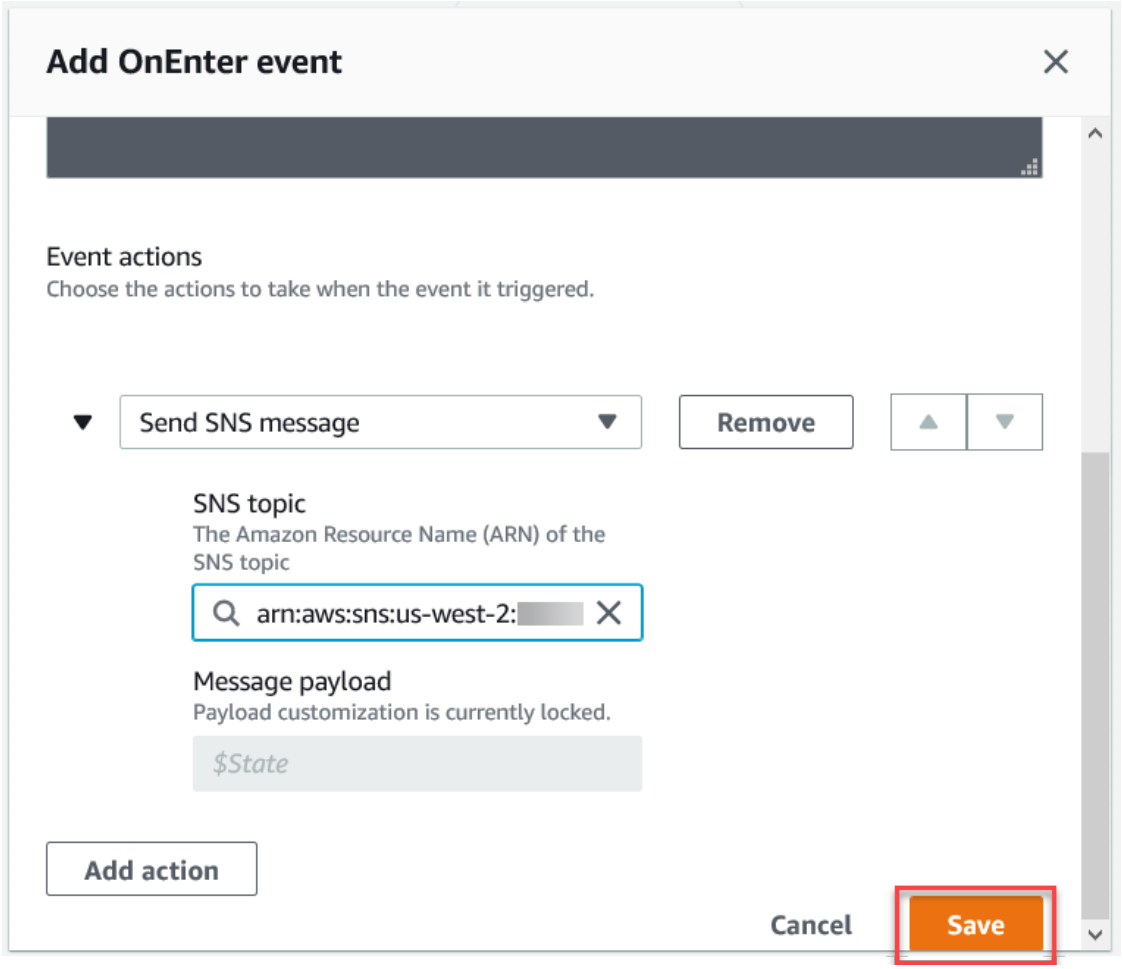
15. 选择第二个状态untitled_state_1.
16. 在州窗格中，输入州/省而且对于Enter，选择添加活动.



17. 在Add OnEnter 事件页面，输入Event name (事件名称),事件状况并且，选择添加操作.



18. 对于选择一个动作，选择发送 SNS 消息.
    a. 对于SNS 主题，输入您的 SNS 主题的目标 ARN。
    b. 选择Save（保存）。



19. 继续在示例中添加事件。

    a. 对于OnInput，选择添加活动，然后输入并保存以下事件信息。

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

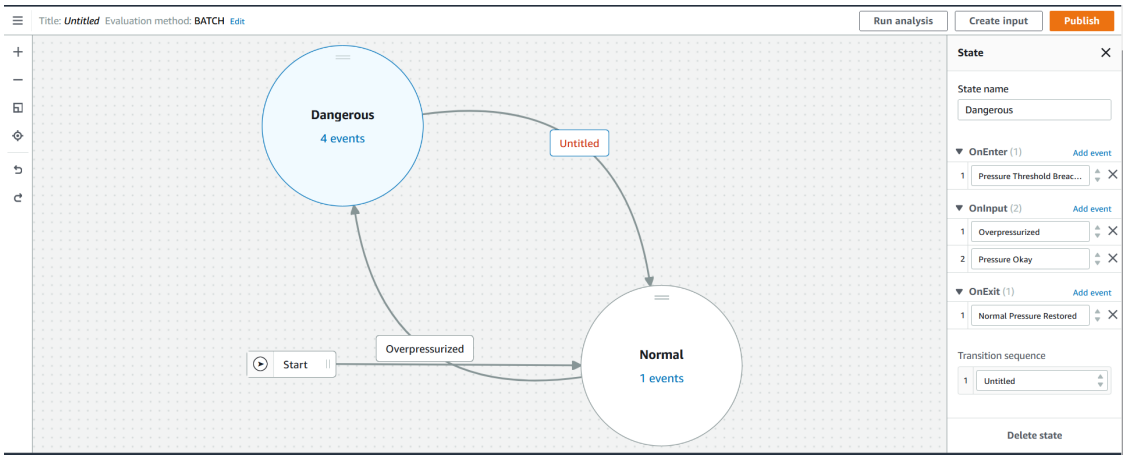
    b. 对于OnInput，选择添加活动，然后输入并保存以下事件信息。

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

    c. 对于OnExit，选择添加活动，然后使用您创建的 SNS 主题的 ARN 输入和保存以下事件信息。

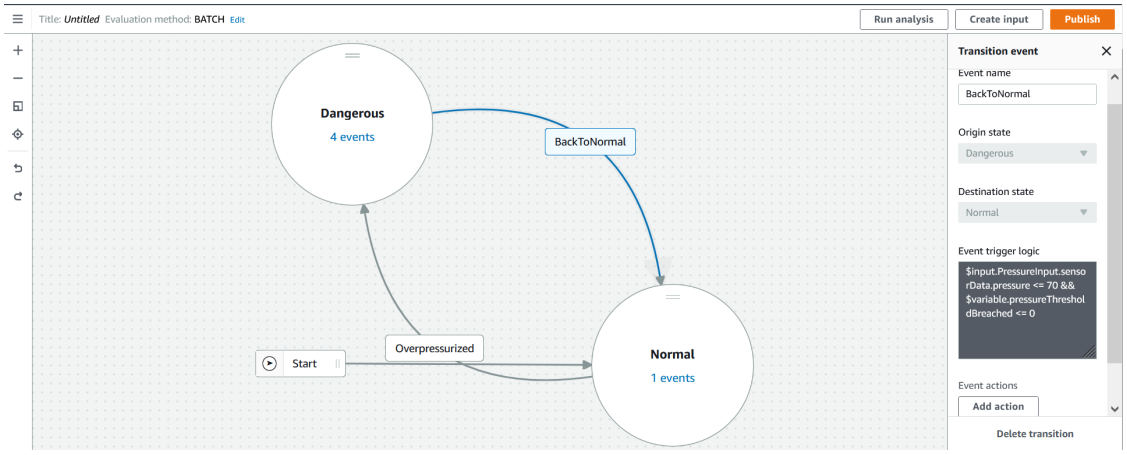
```
Event name: Normal Pressure Restored
Event condition: true
Event actions:
  Send SNS message:
    Target arn: arn:aws:sns:us-east-1:123456789012:pressureClearedAction
```

20. 在第二个状态下暂停 (危险)。州周上会出现一个箭头
21. 单击箭头并将其从第二个状态拖动到第一个状态。带标签的定向线无标题出现。



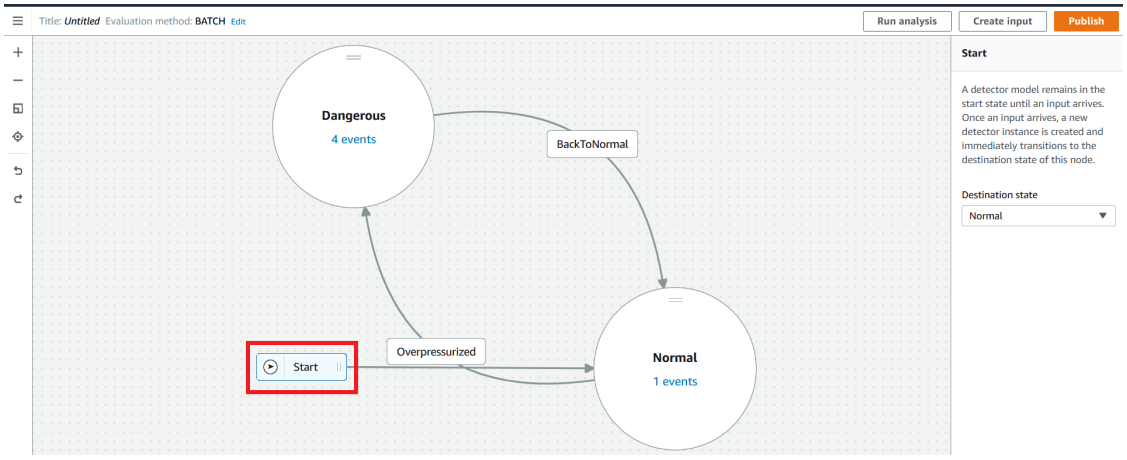
22. 选择无标题line and in the过渡活动窗格，输入Event name (事件名称)和事件触发逻辑使用以下信息。

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

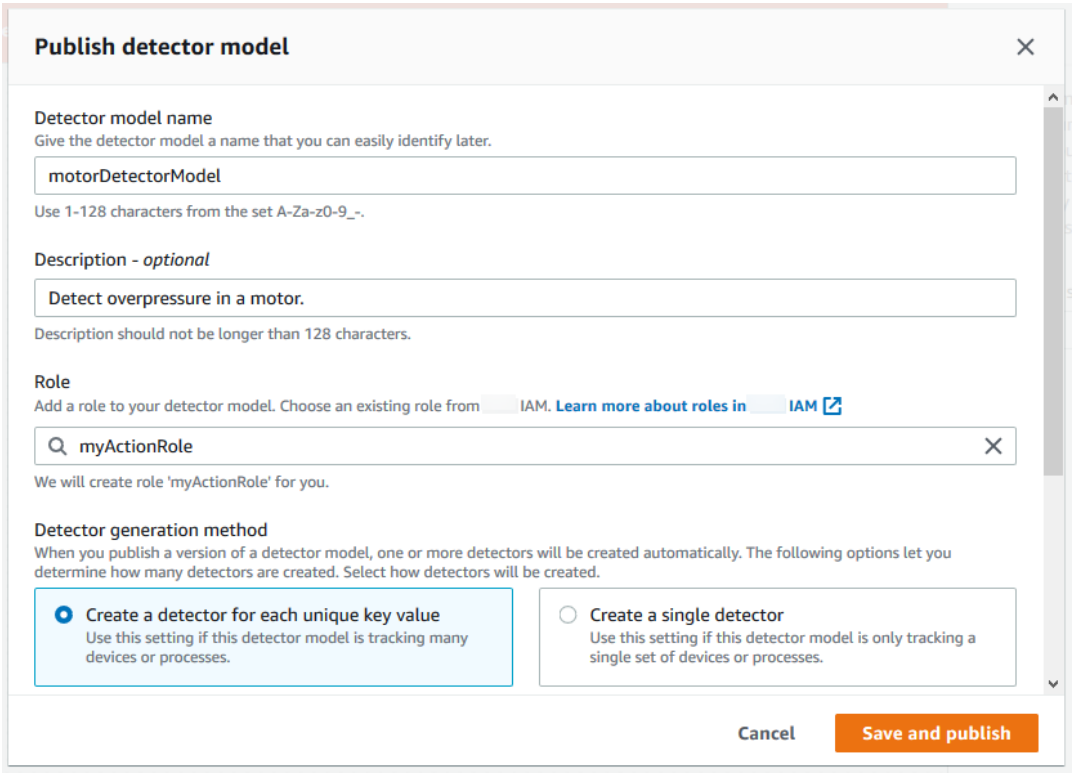


有关我们为何要进行测试的更多信息`$input`值和`$variable`触发器逻辑中的值，请参阅中变量值的可用性条目探测器模型限制 (p. 61).

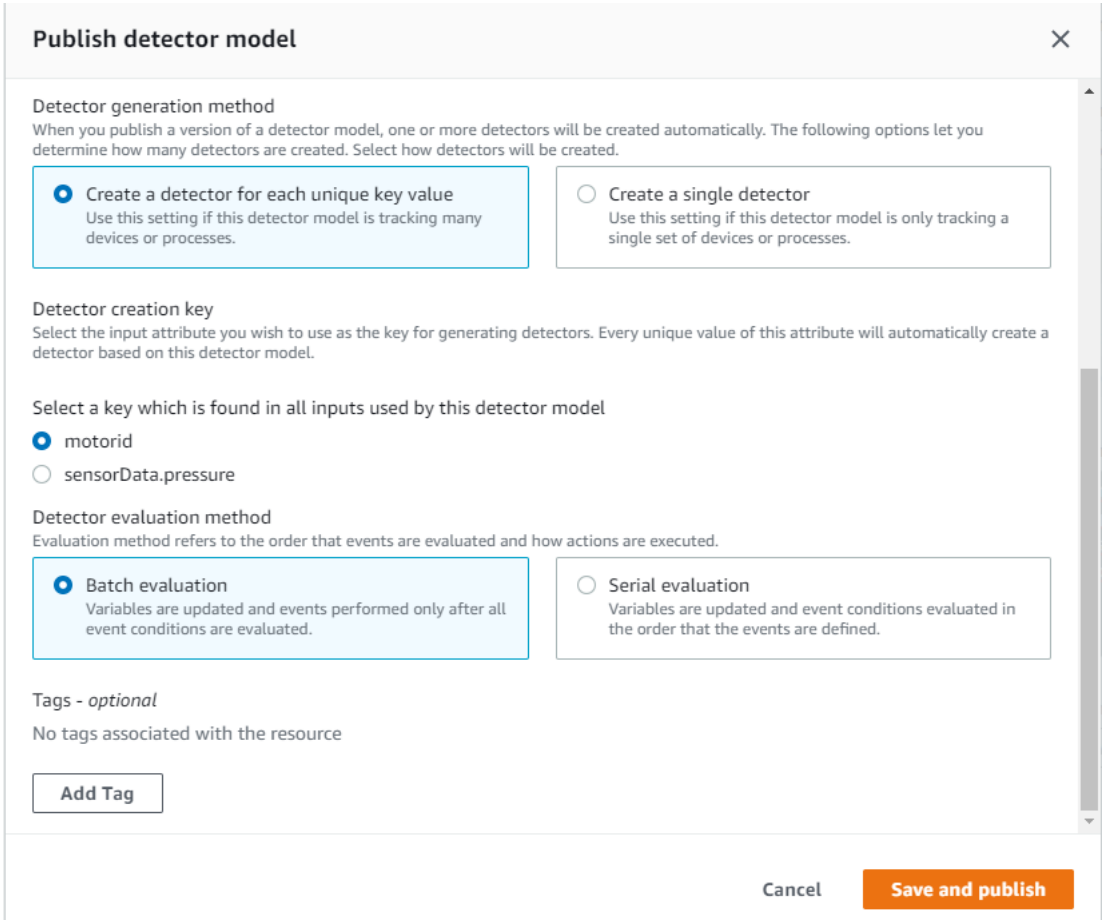
23. 选择启动状态。默认情况下，此状态是在您创建探测器模型时创建的）。在启动窗格中，选择目的地州/省（例如，普通)。



24. 接下来，将您的探测器模型配置为监听输入。在右上角，选择发布.
25. 在发布探测器模型页面上，请执行以下操作。
    a. 输入探测器模型名称，一个说明，以及 a 的名字Role. 将为您创建此角色。
    b. 选择为每个唯一的密钥值创建一个检测器. 创建和使用你自己的Role，按中的步骤操作 使用 IAM 控制台管理角色和权限 (p. 5)并将其作为Role此处。



26. 对于探测器创建密钥，请选择您先前定义的输入的其中一个属性的名称。您选择作为检测器创建密钥的属性必须存在于每个消息输入中，并且对于发送消息的每台设备而言，该属性必须是唯一的。此示例使用motorid属性。
27. 选择保存并发布。



您可以创建探测器模型定义的备份副本（采用 JSON 格式），重新创建或更新探测器模型，或者将其用作模板来创建另一个探测器模型。

您可通过控制台或使用以下 CLI 命令完成这一操作。如有必要，更改探测器模型的名称，使其与您在上一步中发布时使用的名称相匹配。

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

这会创建一个文件 (`motorDetectorModel.json`)，内容类似以下内容。

```
{
    "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"
        }
    }
}
```

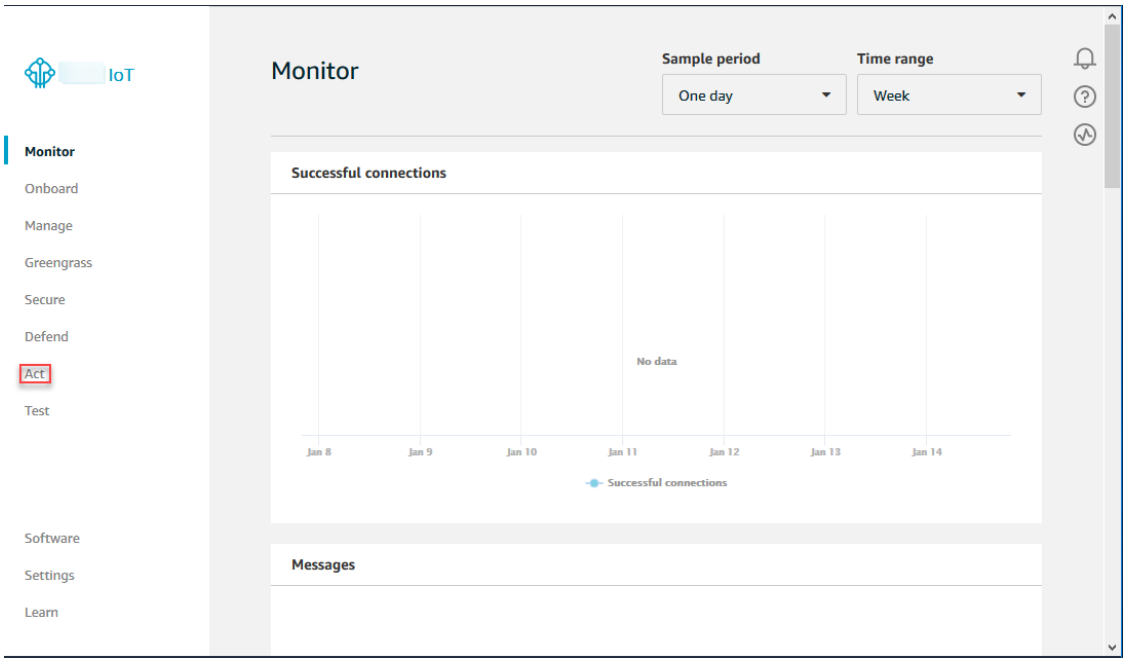
发送输入以测试探测器模型

在中可通过多种方式接收遥测数据Amazon IoT Events（请参阅支持的操作 (p. 87))。本主题说明了如何创建Amazon IoT规则在里面Amazon IoT控制台将消息作为输入转发给你的Amazon IoT Events探测器。您

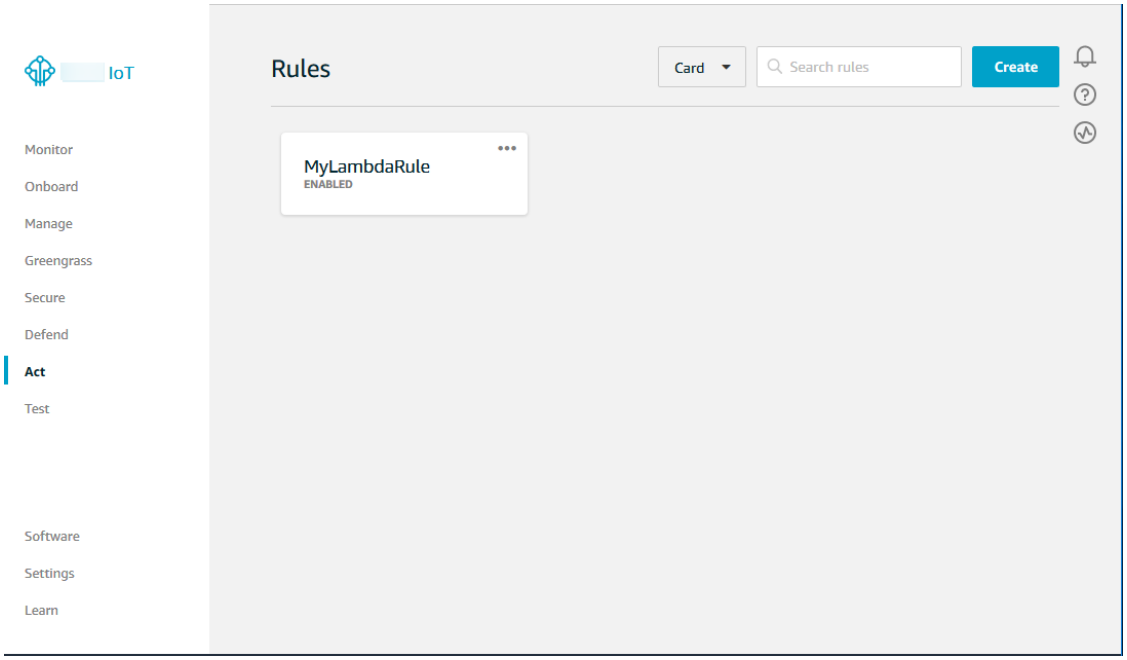
可以使用Amazon IoT控制台的 MQTT 客户端发送测试消息。您可以使用此方法将遥测数据导入Amazon IoT Events当您的设备能够使用发送 MQTT 消息时Amazon IoT消息代理。

发送输入以测试探测器模型

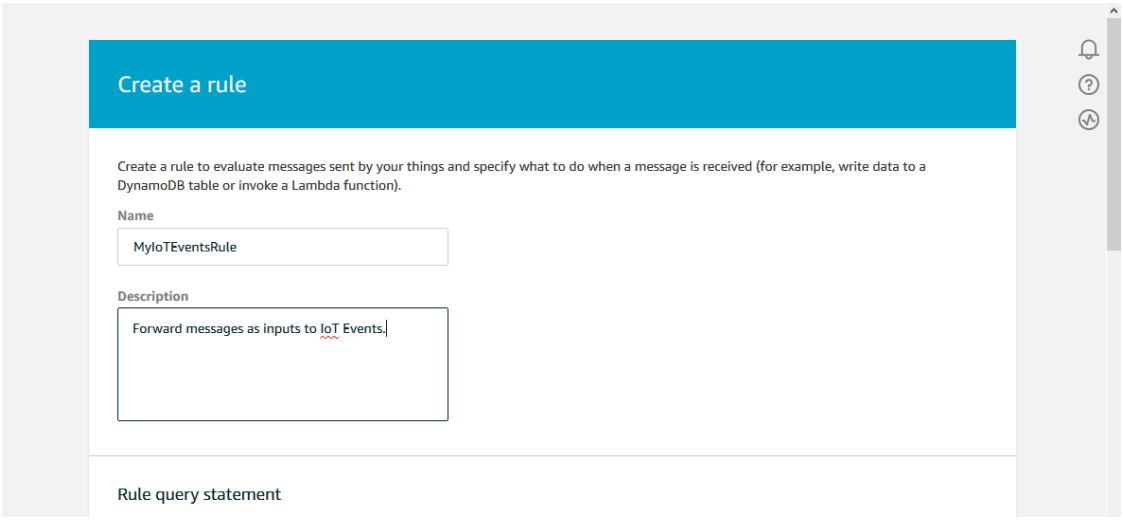
1. 打开AmazonIoT 核心控制台. 在导航窗格中，选择法案.



2. 在 Rules 页面，选择 Create。

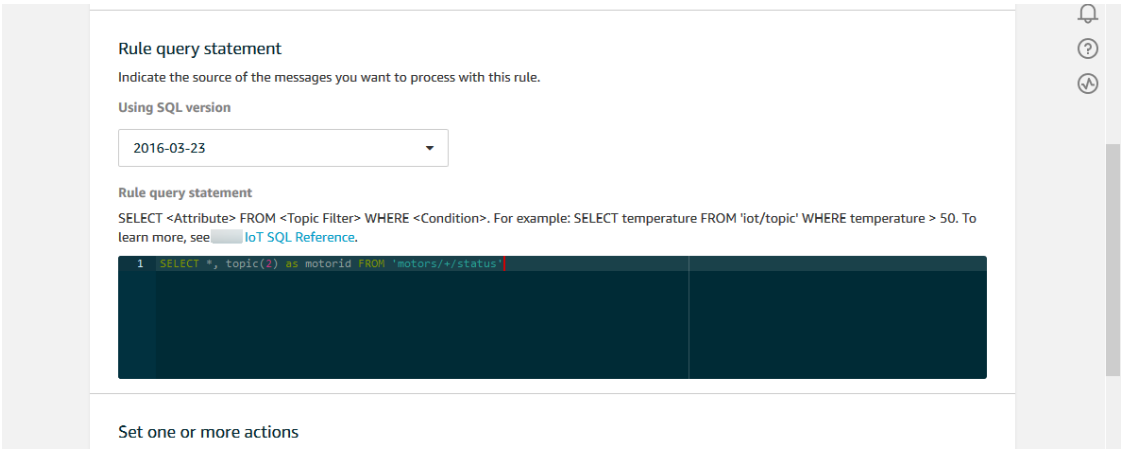


3. 在创建规则页面，输入名称和说明.

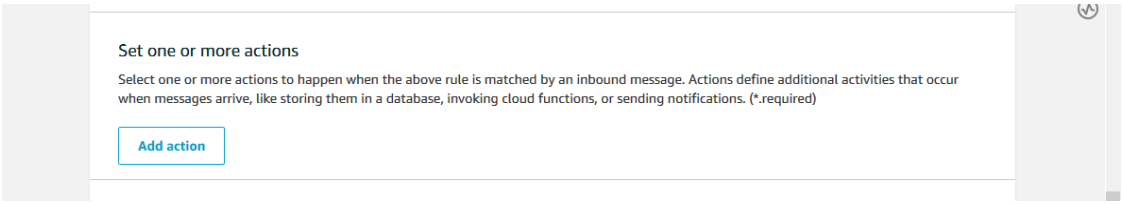


4. 在规则查询语句，输入以下内容。

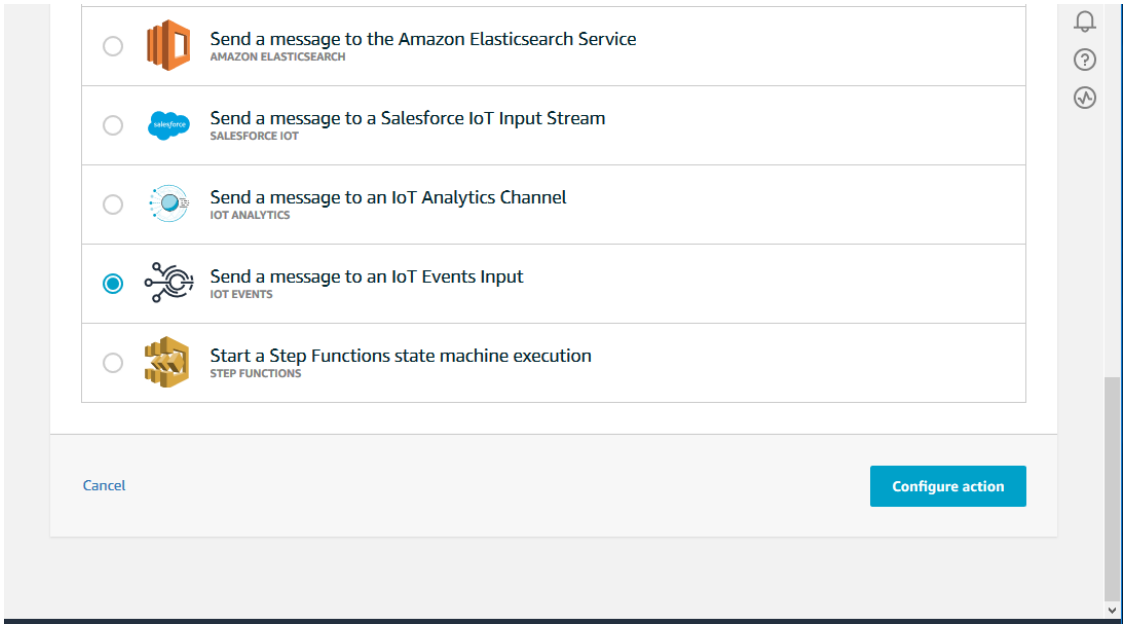
```
SELECT *, topic(2) as motorid FROM 'motors/+/status'
```



5. 在设置一个或多个操作中，选择添加操作。



6. 在选择一项操作页面上，选择将消息发送到Amazon IoT Events输入然后选择配置动作.

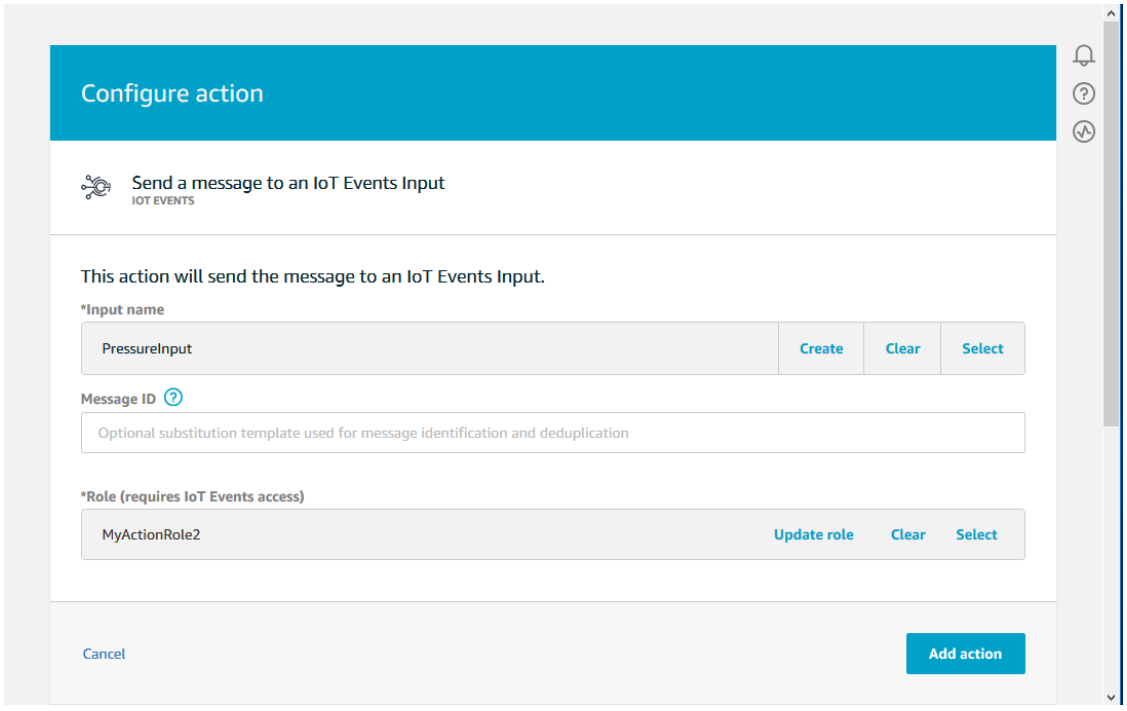


7. 在配置操作页面上，执行以下操作：

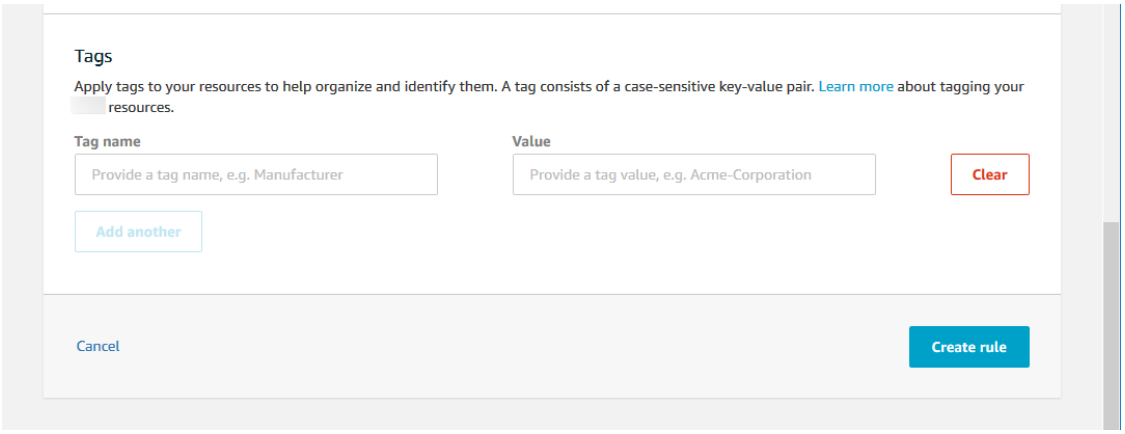
   a. 对于输入名称，输入您在上一节中创建的名称。

   b. 对于Role，选择创建角色而在创建一个新角色窗口，输入名称然后选择创建角色. 这将创建一个有权将消息转发到的角色Amazon IoT Events.

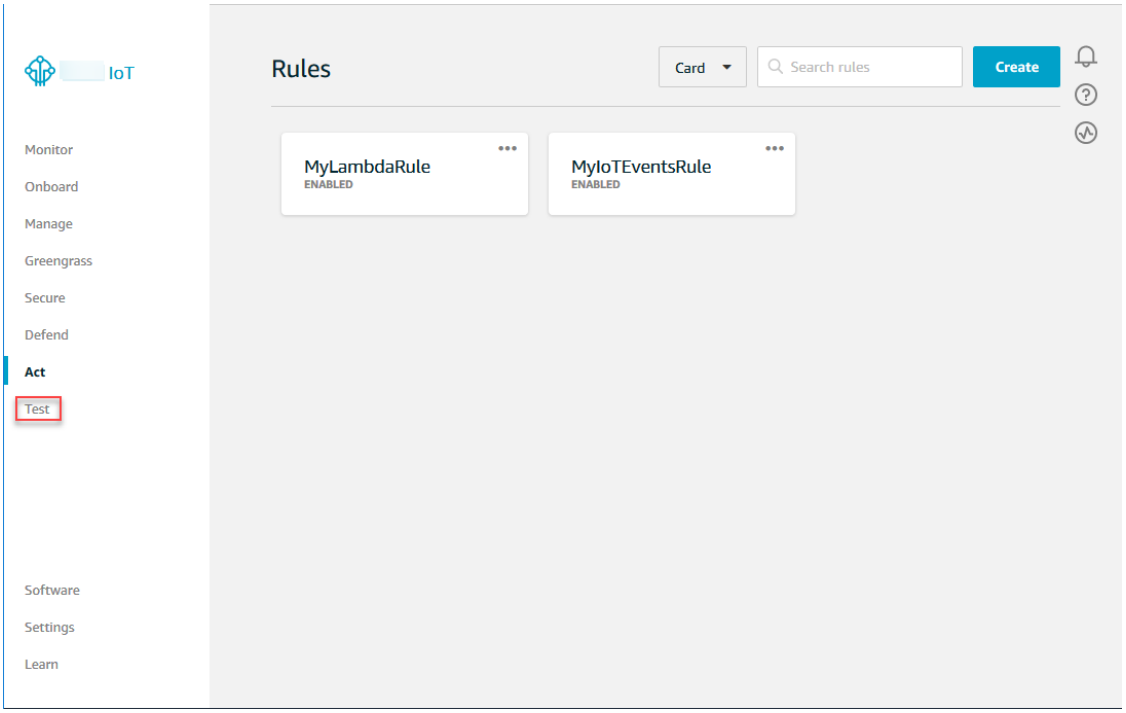
   c. 回来了配置动作页面，选择添加操作.



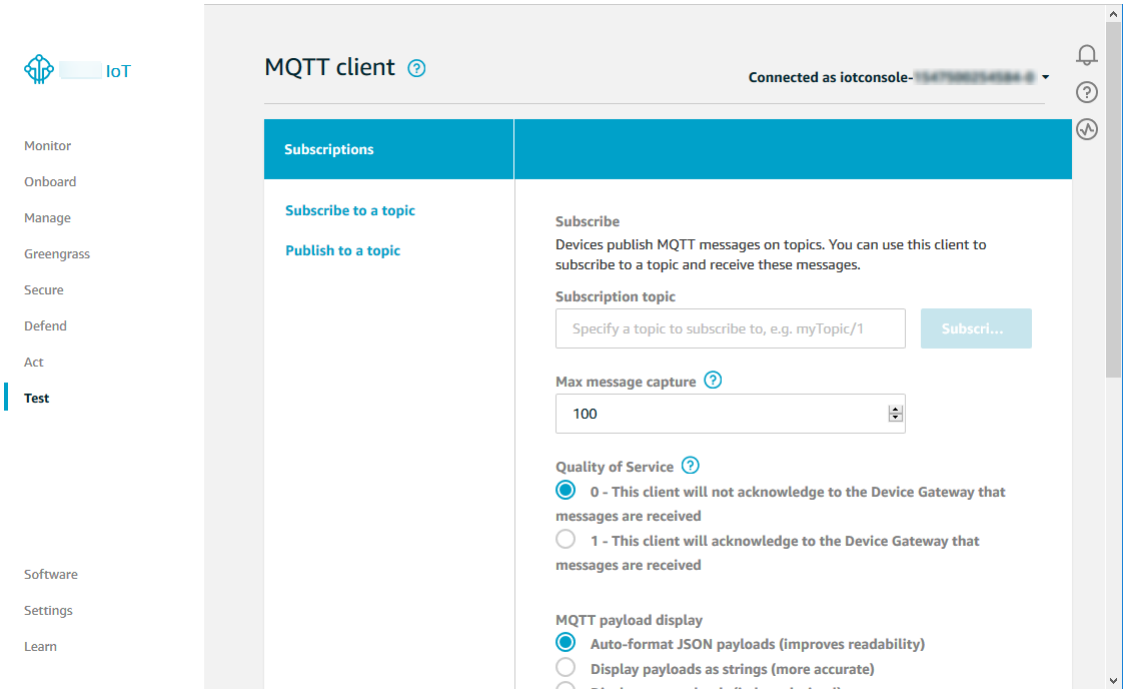
8. 在创建规则页面，选择创建规则.



9. 在Rule页面，在导航窗格中，选择测试.

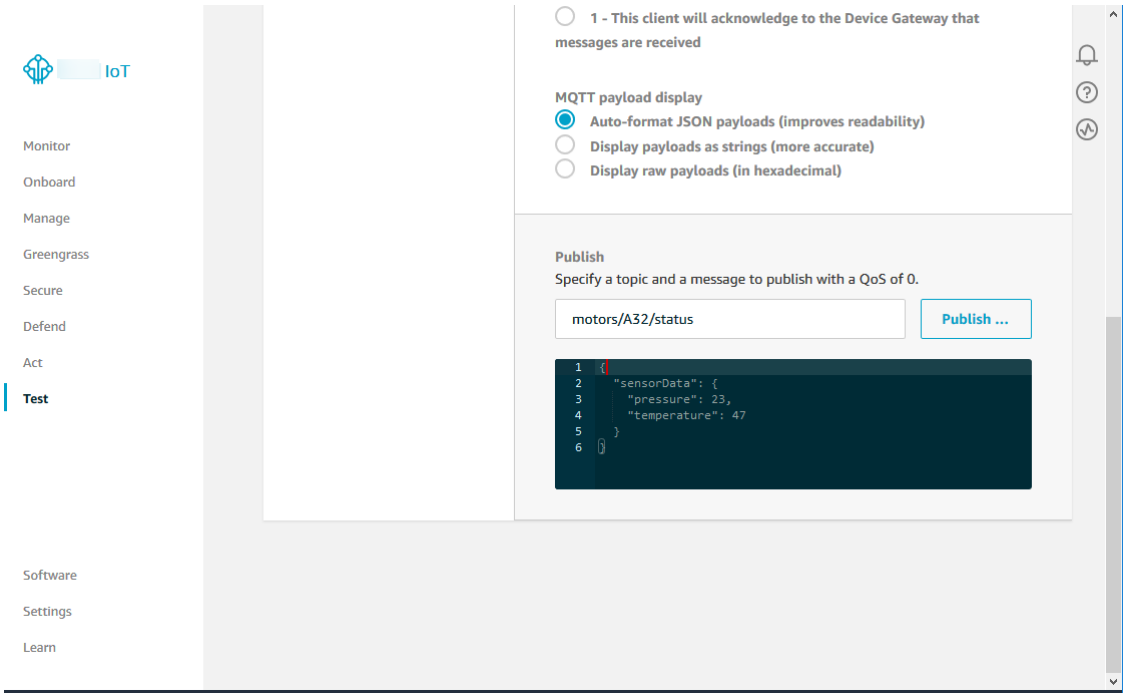


10. 在MQQTT 客户端页面，选择向主题发布.

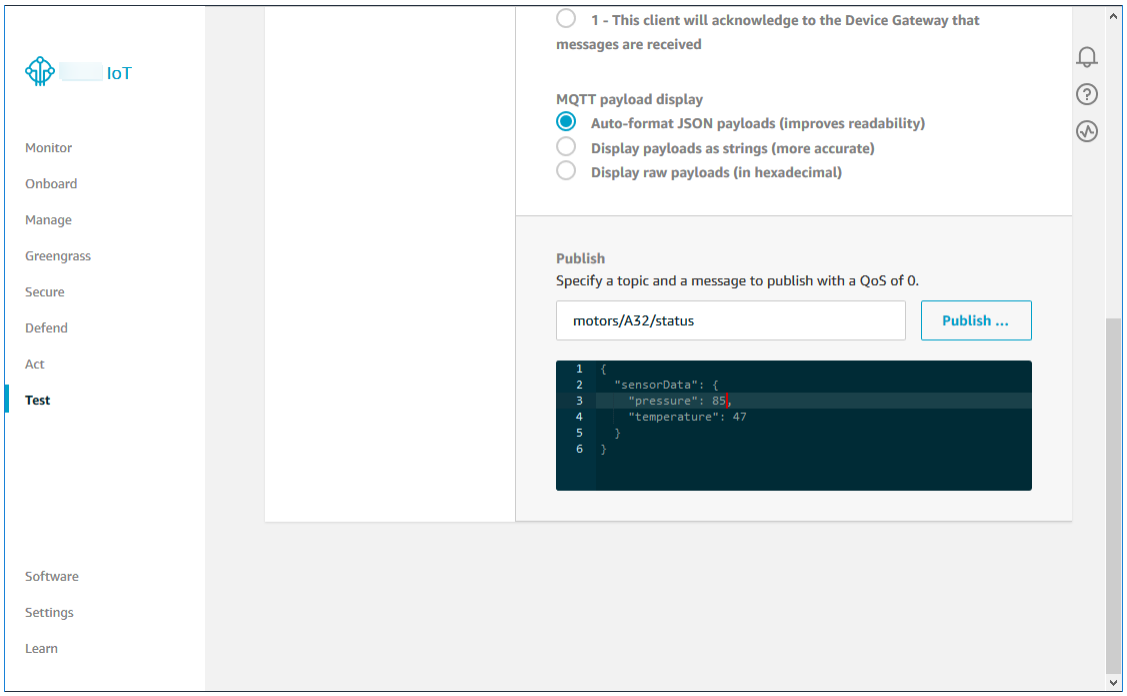


11. 在发布部分，输入主题，在编辑器中输入以下负载，然后选择发布.

```
{
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```



12. 对于发布，保持话题不变，但要更改"pressure"在有效载荷中变为一个大于您在探测器模型中指定的阈值的值（例如**85**）。



13. 选择 Publish。

您创建的检测器实例会生成并向您发送一条 SNS 消息。继续发送压力读数高于或低于压力阈值（本例中为 70）的消息，以查看探测器的运行情况。

在本示例中，您必须发送三条压力读数低于阈值的消息才能过渡回到普通状态并接收 SNS 消息，表示超压状态已清除。一旦回来了普通状态，一条压力读数超过极限的消息会导致探测器进入危险陈述并发送一条 SNS 消息以表明该情况。

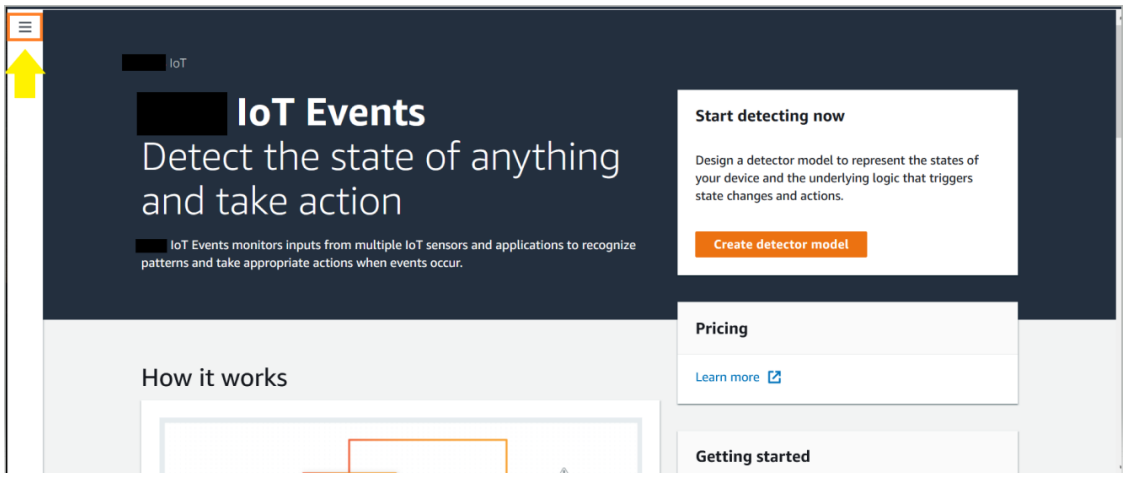
现在您已经创建了一个简单的输入和探测器模型，请尝试以下操作。

- 在控制台上查看更多探测器模型示例（模板）。
- 按中的步骤操作简单ile step-by-step 示例 (p. 55)使用创建输入和探测器模型Amazon CLI
- 了解详情表达式 (p. 97)在活动中使用。
- 了解 支持的操作 (p. 87)。
- 如果有些东西不起作用，请参阅排除 Amazon IoT Events 的故障 (p. 189).

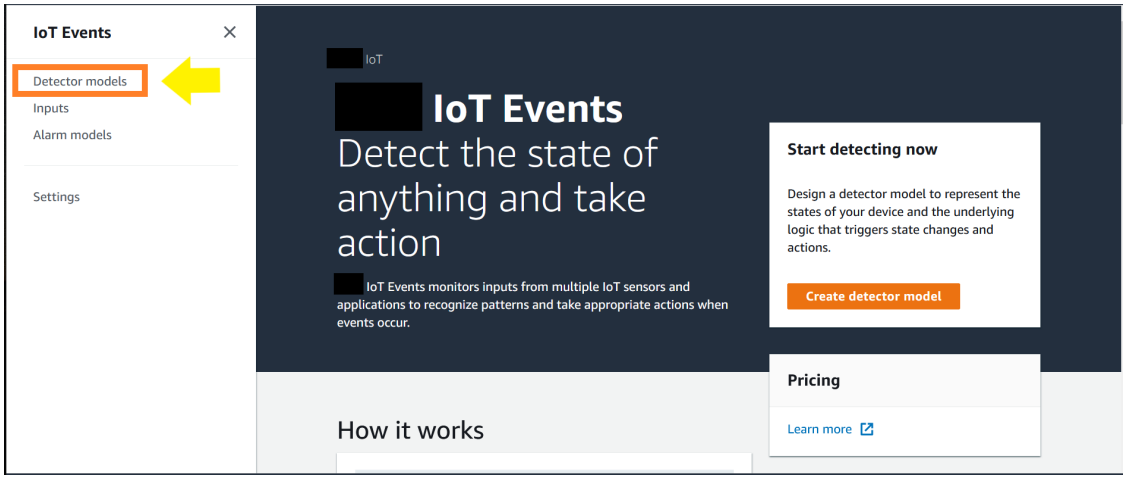
删除检测器

本主题说明了如何删除一个或多个。探测器. 您可以使用这些步骤清理所有未使用或过时的探测器。

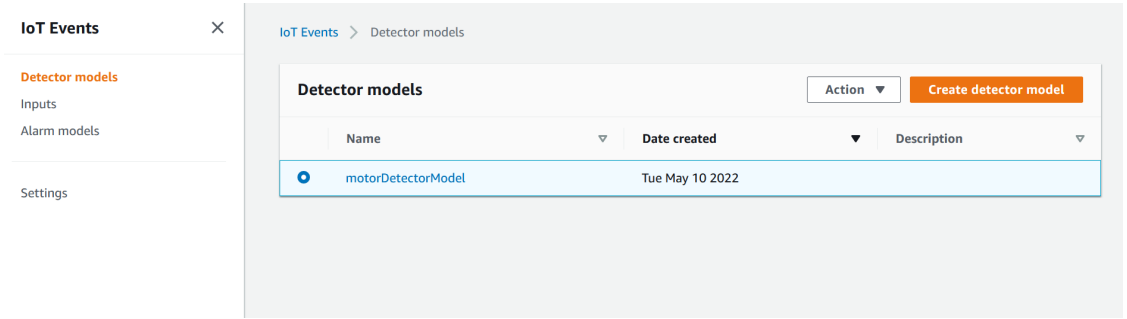
1. 打开 Amazon IoT Events 控制台。
2. 要展开导航窗格，请选择菜单。



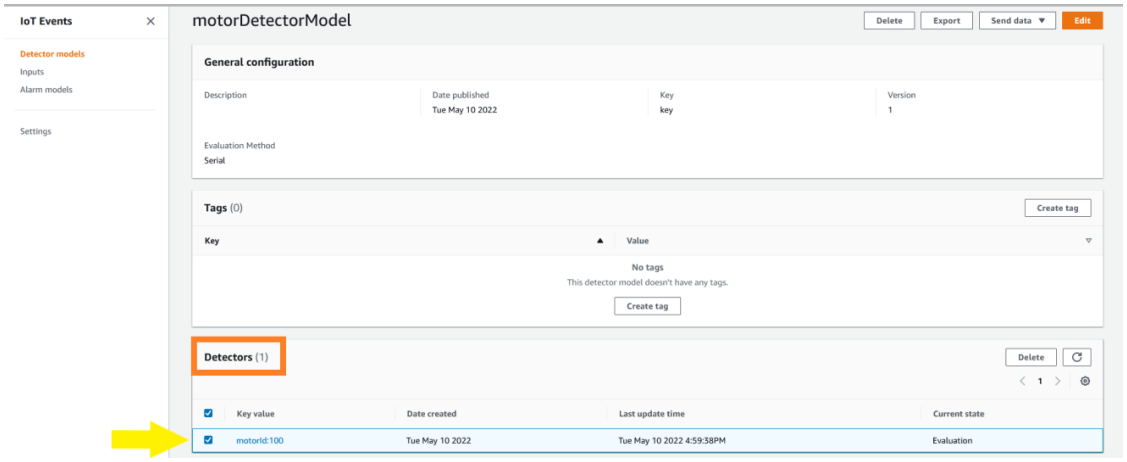
3. 在导航窗格中，选择探测器模型.



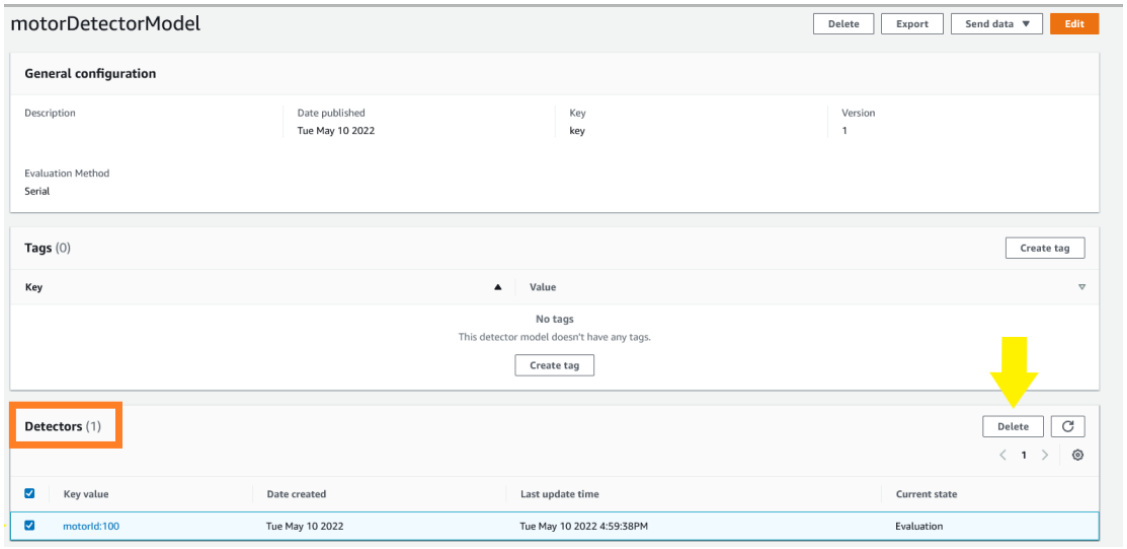
4. 选择的名称探测器模型对于要删除的探测器。



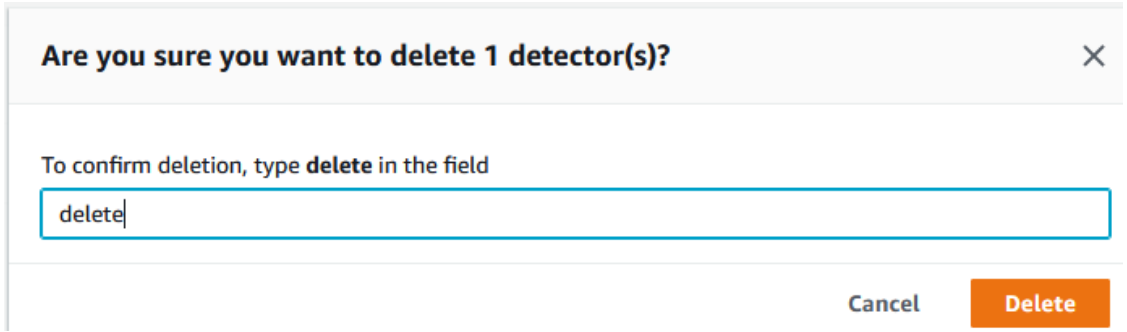
5. 在探测器部分，找到要删除的探测器名称并选中其旁边的复选框。



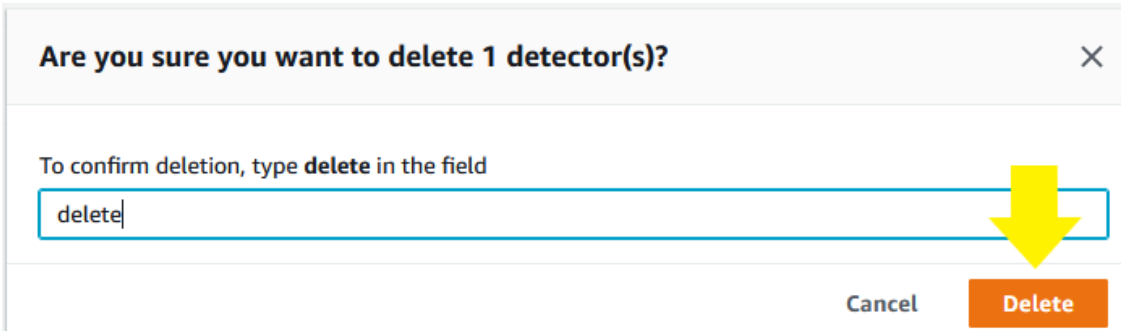
6. 还在探测器部分，选择Delete.



7. 在出现的对话框中，输入以下命令以确认您要删除选定的检测器删除在文本框中。



8. 选择 Delete（删除）。


Are you sure you want to delete 1 detector(s)?
To confirm deletion, type delete in the field
delete
Cancel
Delete

Amazon IoT Events 的最佳实践

按照这些最佳实践可以从中获得最大的好处Amazon IoT Events.

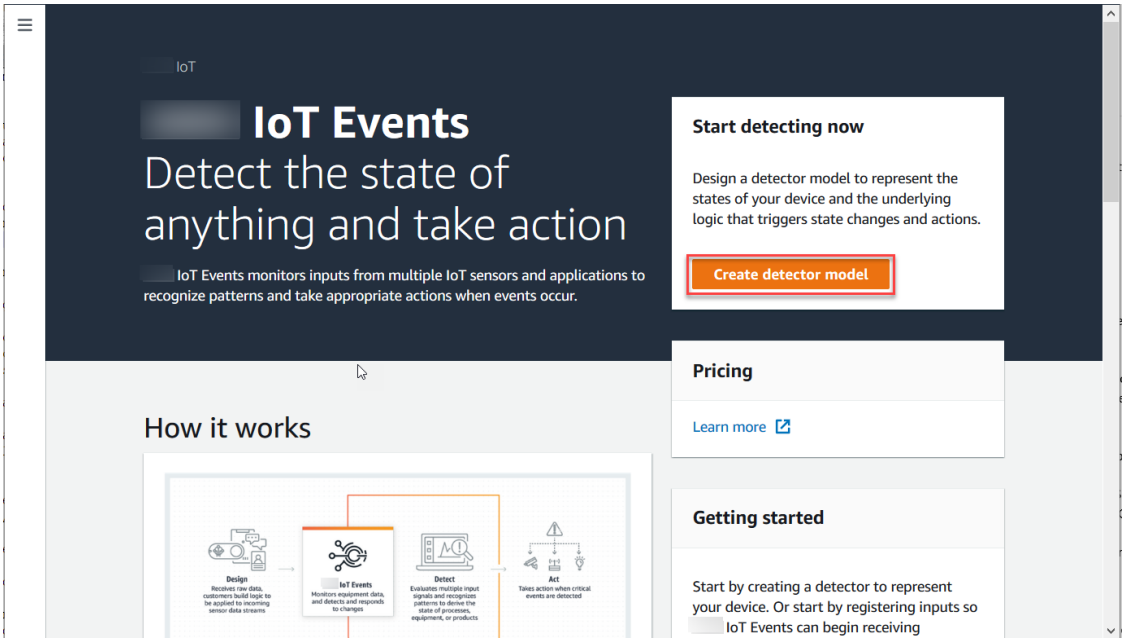
主题

启用Amazon CloudWatch 开发时记录Amazon IoT Events探测器模型

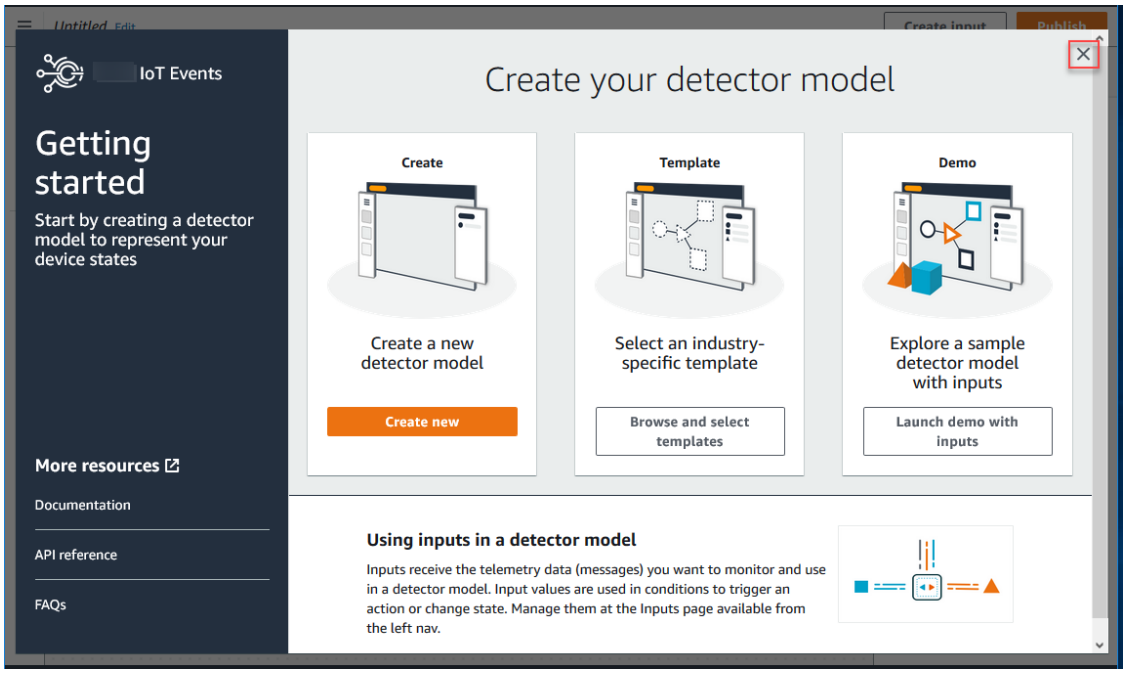
亚马逊 CloudWatch 监控Amazon资源以及您在上运行的应用程序Amazon实时。与 CloudWatch，您了解系统范围的资源使用情况、应用程序性能和运行状况。当你开发或调试时Amazon IoT Events探测器模型，CloudWatch帮你知道什么Amazon IoT Events正在做，以及它遇到的任何错误。

启用 CloudWatch

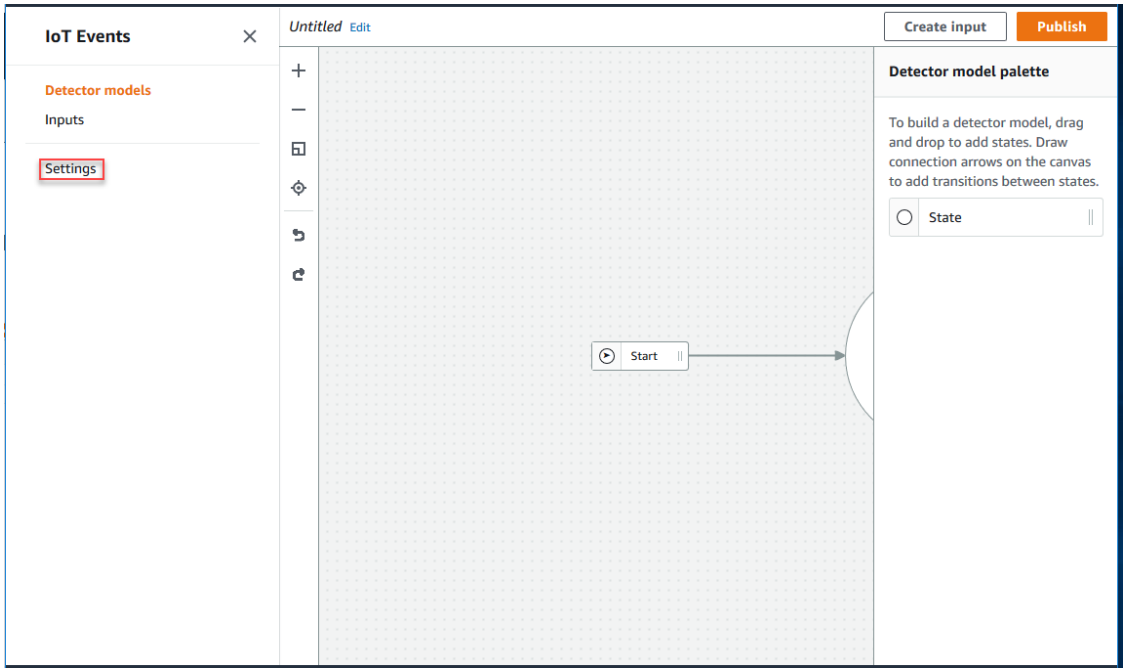
1. 如果您尚不了解，请按照中的步骤进行操作设置 Amazon IoT Events 的权限 (p. 3)创建带有附加策略的角色，该策略授予创建和管理权限 CloudWatch 的日志Amazon IoT Events.
2. 在Amazon IoT Events控制台，选择左上角的菜单图标打开导航窗格。



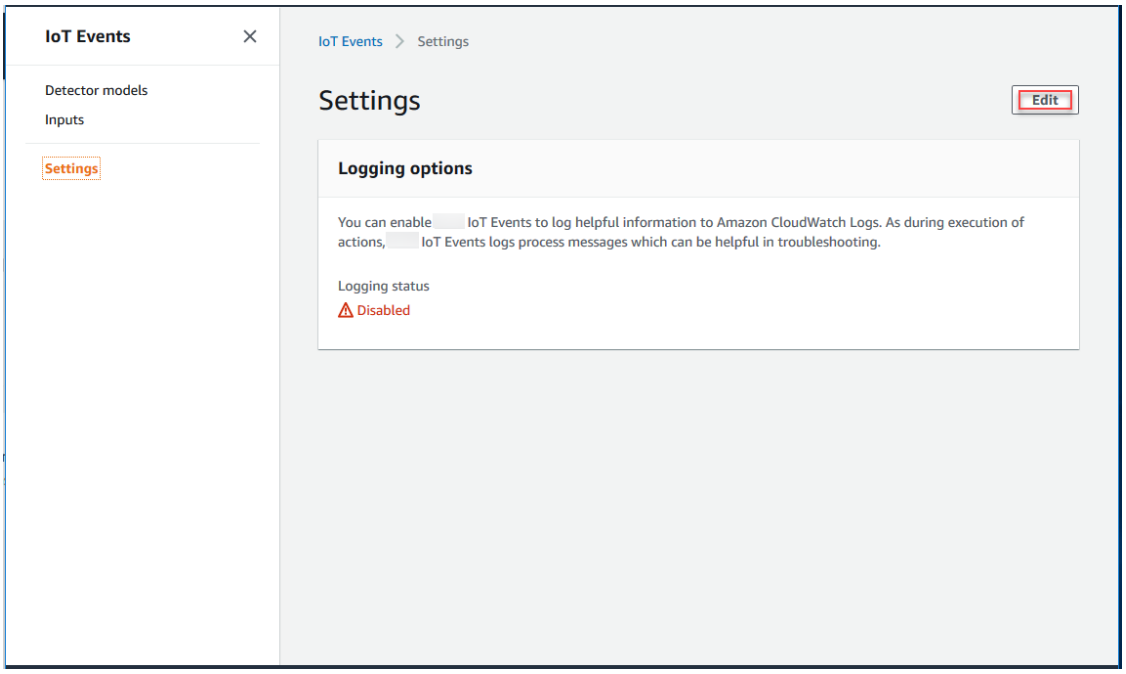
如果您在入门页面，选择X在右上角关闭该页面然后转到探测器模型调色板. 选择左上角的菜单图标。



3. 在导航窗格中，选择 Settings (设置)。



4. 在设置页面，选择编辑.



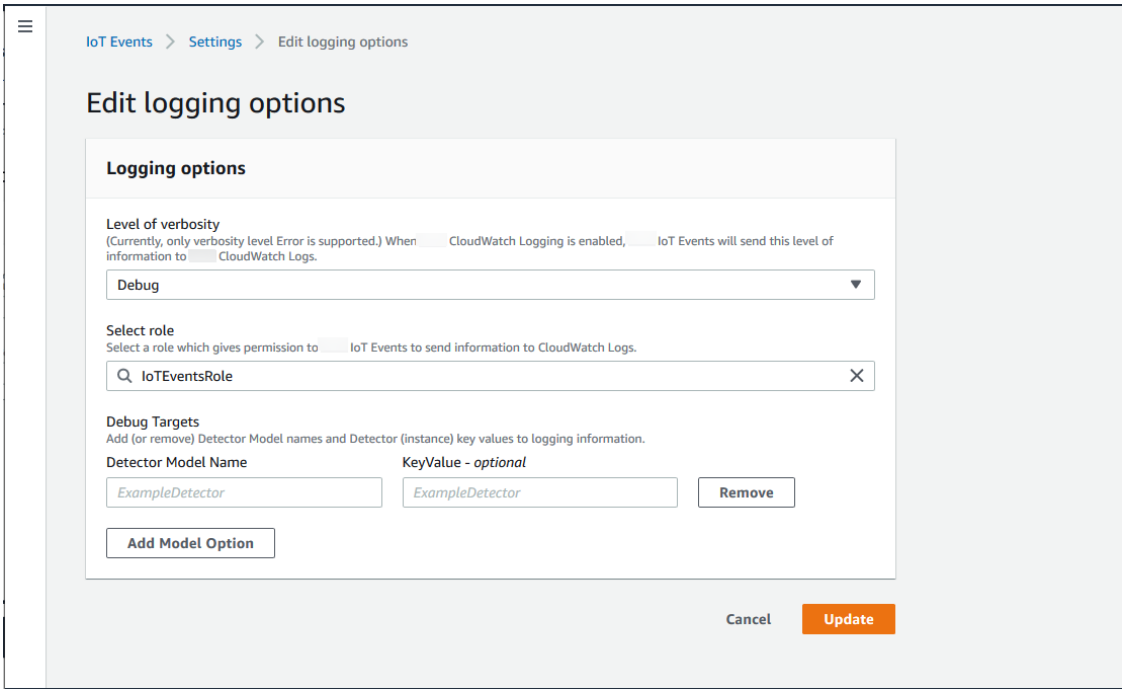
5. 在编辑日志记录选项页面上，请执行以下操作：

   a. 选择详细程度.

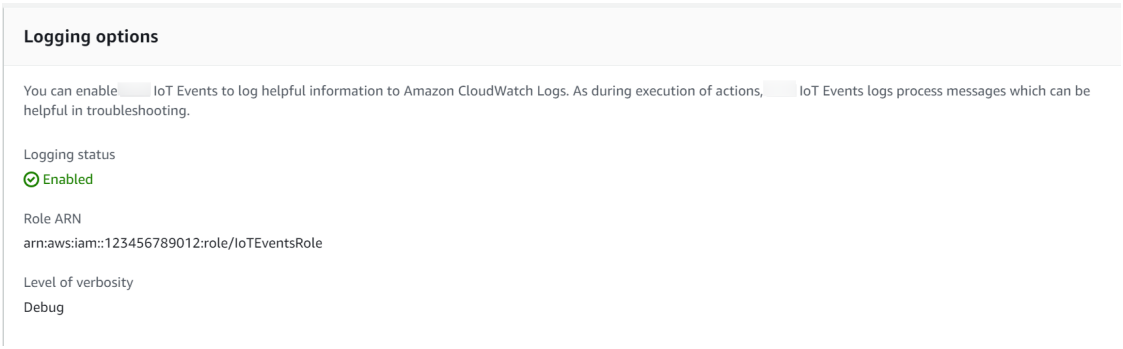
   b. 对于选择角色，请选择一个具有足够权限的角色来执行您选择的日志操作。

   c. 如果你选择了Debug对于详细程度，你也可以选择添加模型选项并添加一个探测器模型名称和（可选）KeyValue指定要记录的探测器模型和特定探测器（实例）。

   d. 选择 Update（更新）。



您的日志记录选项已成功更新。



定期发布，以便在探测器中工作时保存您的探测器模型Amazon IoT Events控制台

当您使用以下应用程序时：Amazon IoT Events控制台，您正在进行的工作保存在浏览器中。但是，你必须选择发布将您的探测器模型保存到Amazon IoT Events. 发布探测器模型后，您发布的作品将在您用于访问账户的任何浏览器中可用。

Note

如果你不发布你的作品，它将不会被保存。发布探测器模型后，便无法再更改其名称。但是，您可以继续修改其定义。

存储你的Amazon IoT Events数据，以避免由于长时间不活动而可能丢失数据

如果你不使用Amazon IoT Events在很长一段时间内，您的数据（包括探测器模型）可能会被自动删除。例如，很长一段时间可能意味着你不会产生费用，也不会创建探测器模型。但是，如果没有至少提前 30 天通知您，我们不会删除数据或探测器模型。如果您需要在较长时间内存储数据，请考虑使用Amazon存储服务.

教程

本章将向您介绍如何：

- 获取帮助，以决定在探测器模型中包含哪些状态，并确定您需要一个探测器实例还是多个探测器实例。
- 请看一个使用Amazon CLI.
- 创建输入以接收来自设备的遥测数据，并创建探测器模型以监控和报告发送该数据的设备的状态。
- 查看输入、探测器模型和Amazon IoT Events服务。
- 查看更复杂的探测器模型示例，其中包含注释。

主题

使用Amazon IoT Events监控 IoT 设备

您可以使用Amazon IoT Events监控您的设备或进程，并根据重大事件采取行动。为此，请按照以下基本步骤操作：

创建输入

您必须具有将遥测数据导入设备和流程的方法Amazon IoT Events. 您可以通过以下方式发送消息来实现此目的输入到Amazon IoT Events. 可以通过多种方式将消息作为输入发送：

- 使用 BatchPutMessage操作。
- 定义一个`iotEvents`规则操作对于Amazon IoT Core规则引擎. 规则操作会将您输入的消息数据转发到Amazon IoT Events.
- 中Amazon IoT Analytics，使用CreateDataset操作以创建数据集`contentDeliveryRules`. 这些规则指定了Amazon IoT Events自动发送数据集内容的输入。
- 定义一个IotEvents在一个Amazon IoT Events探测器模型`onInput`,`onExit`要么`transitionEvents`事件。有关检测器模型实例和启动操作的事件的信息将以您指定的名称的输入形式反馈到系统中。

在您的设备开始以这种方式发送数据之前，必须定义一个或多个输入。为此，请为每个输入指定一个名称，并指定输入监视传入消息数据中的哪些字段。Amazon IoT Events以 JSON 负载的形式从多个来源接收其输入。每个输入可以单独操作，也可以与其他输入组合以检测更复杂的事件。

创建探测器模型

定义一个探测器模型（您的设备或进程的模型）状态. 对于每种状态，您定义条件（布尔值）逻辑，该逻辑评估传入的输入以检测重要事件。检测到事件后，它可以使用其他方法更改状态或启动自定义或预定义的操作Amazon服务。您可以定义其他事件，这些事件将在进入或退出某个状态以及满足某个条件（可选）时启动操作。

在本教程中，当模型进入或退出特定状态时，您将发送 Amazon SNS 消息作为操作。

监控设备或进程

如果您正在监视多个设备或进程，则可以在每个输入中指定一个字段来标识输入来自的特定设备或进程。（请参阅“文件”`key`字段中的子位置类型`CreateDetectorModel`。）当识别出一台新设备时（在

由该设备标识的输入字段中看到一个新值（key），则会创建一个探测器。（每个探测器都是探测器模型的实例。）然后，新的探测器继续响应来自该设备的输入，直到其探测器模型被更新或删除。

如果你在监控单个进程（即使有多个设备或子进程正在发送输入），则无需指定唯一标识key字段中返回的子位置类型。在这种情况下，会在第一个输入到达时创建单个检测器（实例）。

将消息作为输入发送到您的探测器模型

有多种方法可以将来自设备或进程的消息作为输入发送到Amazon IoT Events检测器，不需要您对消息进行其他格式化。在本教程中，您将使用Amazon IoT控制台写一个Amazon IoT Events行动规则为Amazon IoT Core规则引擎，用于将您的消息数据转发到Amazon IoT Events. 为此，您需要按名称标识输入。然后您继续使用Amazon IoT控制台生成一些消息，这些消息作为输入转发给Amazon IoT Events.

你怎么知道在探测器模型中你需要哪些状态？

要确定您的探测器模型应处于哪些状态，请首先确定您可以采取哪些操作。例如，如果你的汽车使用汽油，你可以在开始旅行时查看燃油表，看看是否需要加油。在这里，你有一个动作：告诉司机“去加油”。您的探测器模型需要两种状态：“汽车不需要燃料”和“汽车确实需要燃料”。通常，您要为每个可能的操作定义一个状态，在不需要操作时再定义一个状态。即使动作本身更复杂，这也是有效的。例如，你可能想查一下，并附上关于在哪里可以找到最近的加油站或最便宜价格的信息，但是当你发送“去加油站”的消息时，你就会这样做。

要决定接下来要进入哪个状态，请查看输入。输入包含决定应处于什么状态所需的信息。要创建输入，请在设备或进程发送的消息中选择一个或多个字段，以帮助您做出决定。在此示例中，您需要一个输入来告诉您当前的燃油液位（“满载百分比”）。也许你的车向你发送了几条不同的消息，每条消息都有几个不同的字段。要创建此输入，必须选择消息和报告当前气量表液位的字段。为了简单起见，可以硬编码你将要走的行程时长（“到目的地的距离”）；你可以使用你的平均行程时长。你将根据输入进行一些计算（这个百分比满量转换为多少加仑？是指平均行程长度大于您可以行驶的里程（考虑到您的加仑和您的平均“每加仑里程”）。您可以执行这些计算并发送消息事件.

到目前为止，你有两个状态和一个输入。您需要一个处于第一种状态的事件，该事件根据输入执行计算并决定是否进入第二状态。那是一个过渡事件。(transitionEvents处于一个州的onInput事件列表。开启收到一个输入在第一种状态下，事件表演 a过渡到第二个状态，如果事件是condition已满足。）当您进入第二个状态时，您将在进入该状态后立即发送消息。（你使用的是onEnter事件。进入第二个状态时，此事件会发送消息。无需等待另一个输入到达。）还有其他类型的事件，但举一个简单的例子，仅此而已。

其他类型的事件是onExit和onInput. 一旦收到输入并且条件得到满足，onInput事件执行指定的操作。当操作退出其当前状态并且条件得到满足时，onExit事件执行指定的操作。

你遗漏了什么吗？是的，如何回到第一个“汽车不需要燃料”的状态？加满油箱后，输入显示油箱已满。在你的第二个状态中，你需要一个过渡事件回到第一个状态，这个状态是在收到输入时发生的（在第二个状态中）onInput:活动）。如果它的计算结果显示你现在有足够的气体可以到达你想去的地方，它应该会过渡到第一个状态。

这就是基础知识。一些探测器模型会变得更加复杂，因为添加了反映重要输入的状态，而不仅仅是可能的动作。例如，在跟踪温度的探测器模型中可能有三种状态：“正常”状态、“太热”状态和“潜在问题”状态。当温度升高到一定水平以上，但还没有变得太热时，你就会过渡到潜在的问题状态。除非警报在此温度下停留超过 15 分钟，否则您不想发送警报。如果在此之前温度恢复正常，则探测器会恢复到正常状态。如果计时器到期，则探测器会转换到过热状态并发出警报，谨慎起见。你可以使用变量和一组更复杂的事件条件来做同样的事情。但是，实际上使用另一种状态来存储计算结果通常会更容易。

你怎么知道你需要一个探测器实例还是几个实例？

要决定需要多少实例，请问自己“你有兴趣知道什么？”假设你想知道今天的天气怎么样。正在下雨吗（州）？你需要带雨伞吗（行动）？您可以使用一个报告温度的传感器，另一个报告湿度的传感器，以及其他报告气压、风速和风向以及降水的传感器。但是，您必须同时监控所有这些传感器，以确定天气状态（雨、雪、阴天、晴天）和要采取的适当行动（拿起雨伞或涂防晒霜）。尽管传感器数量众多，但您仍需要一个探测器实例来监视天气状态并告知您要采取何种操作。

但是，如果您是您所在地区的天气预报员，则可能有多个此类传感器阵列的实例，它们位于该地区的不同位置。每个地点的人都需要知道该地点的天气如何。在此情况下，您需要检测器的多个实例。每个位置的每个传感器报告的数据必须包含一个您已指定为`key`字段中返回的子位置类型。此字段启用Amazon IoT Events为该区域创建探测器实例，然后在信息继续到达时继续将此信息路由到该探测器实例。再也不会毁掉头发或鼻子晒伤了！

基本上，如果您有一种情况（一个进程或一个位置）需要监控，则需要一个探测器实例。如果您有许多情况（位置、进程）需要监控，则需要多个探测器实例。

简单ile step-by-step 示例

在此示例中，我们称之为Amazon IoT EventsAPI 使用Amazon CLI命令创建检测器，该探测器模拟发动机的两种状态：正常状态和超压状态。

当发动机中测得的压力超过一定阈值时，模型会转换到超压状态，并发送亚马逊Simple Notification Service (Amazon SNS) 消息，提醒技术人员注意这种情况。当压力连续三个压力读数降至阈值以下时，模型返回正常状态并发送另一条 Amazon SNS 消息以确认条件已消失。我们要求在压力阈值以下连续读取三次，以消除在非线性恢复阶段或一次性异常恢复读数时可能出现的过压/正常信息卡顿现象。

以下是创建探测器的步骤概述。

Create输入.

要监控您的设备和流程，它们必须具有将遥测数据导入 Amazon IoT Events 的方法。执行此操作的方法是将消息作为输入发送到 Amazon IoT Events。有几种方式可以实现：

- 使用 BatchPutMessage操作。此方法很简单，但要求您的设备或进程能够访问Amazon IoT Events通过 SDK 或Amazon CLI.
- 中Amazon IoT Core，写入ileAmazon IoT Events行动规则为Amazon IoT Core规则引擎，用于将您的消息数据转发到Amazon IoT Events. 这按名称标识输入。如果您的设备或进程可以或已经在通过以下方式发送消息，请使用此方法Amazon IoT Core. 这种方法通常需要较少的设备计算能力。
- 中Amazon IoT Analytics，使用 CreateDataset操作以创建数据集`contentDeliveryRules`其中指定的是Amazon IoT Events输入，其中数据集内容自动发送。如果您想根据汇总或分析的数据来控制您的设备或进程，请使用此方法Amazon IoT Analytics.

在您的设备能够以这种方式发送数据之前，必须定义一个或多个输入。为此，请为每个输入指定一个名称，并指定输入监视传入消息数据中的哪些字段。

创建探测器模型

创建探测器模型（您的设备或进程的模型）状态. 对于每个状态，定义条件（布尔值）逻辑，该逻辑评估传入的输入以检测重大事件。检测到事件后，它可以使用其他方法更改状态或启动自定义或预定义的操作Amazon服务。您可以定义其他事件，这些事件将在进入或退出某个状态以及满足某个条件（可选）时启动操作。

监控多个设备或进程

如果您正在监视多个设备或进程，并且想要分别跟踪每个设备或进程，请在每个输入中指定一个字段，以标识输入来自的特定设备或进程。请参阅...`key`字段中的子位置类型`CreateDetectorModel`. 当识别出一台新设备时（在由该设备标识的输入字段中看到一个新值）`key`），则创建了一个探测器实例。新的探测器实例继续响应来自该特定设备的输入，直到其探测器模型被更新或删除。您拥有的唯一探测器（实例）与输入中的唯一值一样多`key`字段之间没有不同。

监控单个设备或进程

如果你在监控单个进程（即使有多个设备或子进程正在发送输入），则无需指定唯一标识`key`字段中返回的子位置类型。在这种情况下，会在第一个输入到达时创建单个检测器（实例）。例如，您可能在房屋的每个房间都有温度传感器，但只有一个 HVAC 设备用于加热或冷却整个房屋。因此，即使每个房间居住者都希望自己的投票（输入）占上风，你也只能将其作为一个单一过程进行控制。

将来自设备或进程的消息作为检测器模型的输入发送到检测器模型中

我们描述了将来自设备或进程的消息作为输入发送到设备或进程的几种方法Amazon IoT Events探测器输入输入. 创建输入并构建探测器模型后，即可开始发送数据。

Note

当您创建探测器模型或更新现有模型时，新的或更新的探测器模型需要几分钟时间才能开始接收消息和创建探测器（实例）。如果探测器模型已更新，则在这段时间内，您可能会继续看到基于先前版本的行为。

主题

创建输入以捕获设备数据

例如，假设您的设备使用以下格式发送消息。

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

你可以创建一个输入来捕获`pressure`数据和`motorid`（用于标识发送消息的特定设备）使用以下内容Amazon CLI命令。

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

文件`pressureInput.json`包含以下内容。

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

创建自己的输入时，请记得首先从您的设备或进程中以 JSON 文件形式收集示例消息。您可以使用它们从控制台或 CLI 创建输入。

创建探测器模型以表示设备状态

中创建输入以捕获设备数据 (p. 56)，你创建了一个`input`基于报告马达压力数据的消息。继续本示例，以下是响应电动机超压事件的探测器模型。

你创建了两个状态：”Normal“，和”Dangerous“。每个探测器（实例）都进入”Normal“说明它的创建时间。该实例是在输入具有唯一值时创建的key “motorid“到了。

如果探测器实例收到的压力读数为 70 或更高，则它会进入”Dangerous“声明并发送 Amazon SNS 消息作为警告。如果连续三次输入的压力读数恢复正常（小于 70），则探测器将返回”Normal“声明并发送另一条 Amazon SNS 消息已全部清除。

此示例探测器模型假设您创建了两个 Amazon SNS 主题，其亚马逊资源名称 (ARN) 在定义中显示为"targetArn": "arn:aws:sns:us-east-1:123456789012:underPressureAction"和"targetArn": "arn:aws:sns:us-east-1:123456789012:pressureClearedAction".

有关更多信息，请参阅 。Amazon Simple Note Ge以及，更具体地说，的文档CreateTopic操作中返回的子位置类型Amazon Simple Notifice.

这个例子还假设你已经创建了一个Amazon Identity and Access Management(IAM) 角色具有相应权限。此角色的 ARN 在探测器模型定义中显示为"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole". 按中的步骤操作设置 Amazon IoT Events 的权限 (p. 3)创建此角色并将该角色的 ARN 复制到探测器模型定义中的相应位置。

您可以使用以下方法创建探测器模型Amazon CLI命令。

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

文件"motorDetectorModel.json"包含以下内容。

```
{
  "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"
}
```

将消息作为输入发送到探测器

现在，您已经定义了一个输入，用于识别从设备发送的消息中的重要字段（参见创建输入以捕获设备数据 (p. 56)）。在上一节中，您创建了一个`detector model`它对发动机中的超压事件做出响应（参见创建探测器模型以表示设备状态 (p. 56)）。

要完成此示例，请从设备（在本例中为计算机）发送消息Amazon CLI已安装）作为探测器的输入。

> **Note**
>
> 当您创建探测器模型或更新现有模型时，新的或更新的探测器模型需要几分钟时间才能开始接收消息并创建探测器（实例）。如果您更新探测器模型，在这段时间内，您可能会继续看到基于先前版本的行为。

使用以下Amazon CLI命令发送包含超过阈值的数据的消息。

```
aws iotevents-data batch-put-message --cli-input-json file://highPressureMessage.json
```

文件"`highPressureMessage.json`"包含以下内容。

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

您必须更改`messageId`在发送的每条消息中。如果你不改变它，Amazon IoT Events系统对消息进行重复数据删除。Amazon IoT Events如果消息有相同的消息，则忽略该消息`messageID`作为最近五分钟内发送的另一条消息。

此时，将创建一个探测器（实例）来监视马达的事件`"Fulton-A32"`. 这个探测器进入`"Normal"`注明其创建时间。但是因为我们发送的压力值超过了阈值，所以它会立即转换为`"Dangerous"`状态。这样，检测器会向 ARN 为 Amazon SNS 终端节点发送消息`arn:aws:sns:us-east-1:123456789012:underPressureAction`.

运行以下命令Amazon CLI命令发送包含低于压力阈值的数据的消息。

```
aws iotevents-data batch-put-message --cli-input-json file://normalPressureMessage.json
```

文件normalPressureMessage.json包含以下内容。

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

您必须更改messageId在文件中每次调用BatchPutMessage在五分钟内发出命令。再发送两次消息。消息发送三次后，马达的探测器（实例）"Fulton-A32"向 Amazon SNS 终端节点发送消息"arn:aws:sns:us-east-1:123456789012:pressureClearedAction"然后重新进入"Normal"状态。

> **Note**
>
> 您可以通过以下方式一次发送多条消息BatchPutMessage. 但是，不能保证这些消息的处理顺序。为确保消息（输入）按顺序处理，请一次发送一条，并在每次调用 API 时等待成功响应。

以下是本节中描述的探测器模型示例创建的 SNS 消息有效载荷示例。

在 “压力阈值突破” 事件中

```
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"
}
```

活动中 “恢复正常压力”

```
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"
}
```

如果您定义了任何计时器，则它们的当前状态也会显示在 SNS 消息负载中。

消息有效负载包含有关发送消息时（即运行 SNS 操作时）检测器（实例）状态的信息。您可以使用https://docs.amazonaws.cn/iotevents/latest/apireference/API_iotevents-data_DescribeDetector.html操作以获取有关探测器状态的类似信息。

探测器模型限制

在创建探测器模型时，需要考虑以下事项。

如何使用**actions**领域

这些区域有：actions字段中返回的子位置类型。你可以有多个对象，但每个对象中只允许一个操作。

Example

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

如何使用**condition**领域

这些区域有：condition是否必要transitionEvents并且在其他情况下是可选的。

如果condition字段不存在，它等同于"condition": true.

条件表达式的计算结果应为布尔值。如果结果不是布尔值，则等效于false而且不会启动actions或者过渡到nextState在事件中指定。

变量值的可用性

默认情况下，如果在事件中设置了变量的值，则其新值不可用或用于评估同一组中其他事件的条件。新值不可用或在相同的事件条件中使用onInput,onEnter要么onExit字段中返回的子位置类型。

设置evaluationMethod探测器模型定义中的参数用于更改此行为。当evaluationMethod将设置为SERIAL，变量更新并按事件定义顺序评估事件条件。否则，当evaluationMethod将设置为BATCH或者默认为它，只有在评估了所有事件条件之后，才会更新状态内的变量并执行状态内的事件。

在里面"Dangerous"状态，在onInput字段ile，"$variable.pressureThresholdBreached"在中减去一"Pressure Okay"满足条件时发生的事件（当电流输入的压力小于或等于 70 时）。

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

探测器应过渡回到"Normal"说明何时"$variable.pressureThresholdBreached"达到 0（即，当探测器接收到三个小于或等于 70 的连续压力读数时）。这些区域有："BackToNormal"EventiletransitionEvents必须测试一下"$variable.pressureThresholdBreached"小于或等于 1（不是 0），还要再次验证给出的当前值"$input.PressureInput.sensorData.pressure"小于或等于 70。

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

否则，如果条件仅测试变量的值，则两个正常读数后跟一个超压读数将满足条件并过渡回到"Normal"状态。条件是看这个值"$variable.pressureThresholdBreached"是在上次处理输入时给出的。该变量的值将在中重置为 3"Overpressurized"事件，但请记住，任何人都无法使用这个新值condition.

默认情况下，每次控件进入onInput字段ile，condition只能看到变量在开始处理输入时的值，在中指定的任何操作对其进行更改之前onInput.（情况也是如此onEnter和onExit. 当我们进入或退出状态时对变量所做的任何更改都不适用于同一状态中指定的其他条件onEnter要么onExit字段之间没有不同。

更新探测器模型时的延迟

如果您更新、删除和重新创建探测器模型（请参见UpdateDetectorModel），在删除所有生成的探测器（实例）并使用新模型重新创建探测器之前，会有一段延迟。它们是在新的探测器模型生效和新的输入到达后重新创建的。在这段时间内，输入可能会继续由先前版本的探测器模型生成的探测器处理。在此期间，您可能会继续收到由先前探测器模型定义的警报。

输入键中的空格

输入键中允许有空格，但无论是在输入属性的定义中还是在表达式中引用键值时，对键的引用都必须用反引号括起来。例如，给定一个如下所示的消息负载：

```
{
```

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

使用以下命令定义输入。

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

在条件表达式中，还必须使用反引号来引用任何此类键的值。

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

一个带注释的例子：HVAC 温度控制

以下一些示例 JSON 文件具有内联注释，这使它们成为无效的 JSON。这些示例的完整版本（不含注释）可在以下网址获得暖通空调温度控制 (p. 105).

背景

此示例实现了一个恒温器控制模型，使您能够执行以下操作。

- 仅定义一个可用于监视和控制多个区域的探测器模型。为每个区域创建一个探测器实例。
- 从每个控制区域的多个传感器摄取温度数据。
- 更改某个区域的温度设定点。
- 为每个区域设置操作参数，并在使用实例时重置这些参数。
- 在某个区域动态添加或删除传感器。
- 指定最短运行时间以保护加热和冷却装置。
- 拒绝异常的传感器读数。
- 定义紧急设定点，当任何一个传感器报告温度高于或低于给定阈值时，立即启动加热或冷却。
- 报告异常读数和温度峰值。

输入定义

我们想创建一个探测器模型，用于监视和控制多个不同区域的温度。每个区域可以有多个报告温度的传感器。我们假设每个区域都由一个加热装置和一个冷却装置供电，它们可以打开或关闭以控制该区域的温度。每个区域由一个探测器实例控制。

由于我们监控和控制的不同区域可能具有不同的特征，需要不同的控制参数，因此我们定义了`'seedTemperatureInput'`为每个区域提供这些参数。当我们将这些输入消息之一发送到Amazon IoT Events，创建了一个新的探测器模型实例，它具有我们要在该区域使用的参数。这是该输入的定义。

CLI 命令：

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

文件：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" }
    ]
  }
}
```

响应:

```
{
    "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."
    }
}
```

注意

- 为每个唯一的检测器实例创建一个新的检测器实例'areaId'在任何消息中收到。请参阅...'key'字段中返回的子位置类型'areaDetectorModel'定义。
- 平均温度可能与'desiredTemperature'by the'allowedError'在该区域的加热或冷却装置启动之前。
- 如果有任何传感器报告的温度高于'rangeHigh'，探测器报告峰值并立即启动冷却装置。
- 如果有任何传感器报告的温度低于'rangeLow'，探测器报告峰值并立即启动加热装置。
- 如果有任何传感器报告的温度高于'anomalousHigh'或者低于'anomalousLow'，探测器报告异常的传感器读数，但忽略报告的温度读数。
- 这些区域有：'sensorCount'告诉探测器正在报告该区域有多少传感器。探测器通过为其收到的每个温度读数提供适当的加权系数来计算该区域的平均温度。因此，探测器不必跟踪每个传感器报告的内容，并且可以根据需要动态更改传感器的数量。但是，如果单个传感器离线，则探测器不会知道这一点，也不会留出余地。我们建议您创建另一个探测器模型，专门用于监视每个传感器的连接状态。使用两个互补的探测器模型可以简化两者的设计。
- 这些区域有：'noDelay'值可以是true要么false. 加热或冷却装置开启后，应至少保持开启状态一段时间，以保护设备的完整性并延长其使用寿命。如果'noDelay'将设置为false，探测器实例在关闭冷却和加热装置之前强制延迟，以确保它们在最短的时间内运行。由于我们无法使用变量值来设置计时器，因此在探测器模型定义中对延迟秒数进行了硬编码。

这些区域有：'temperatureInput'用于将传感器数据传输到探测器实例。

CLI 命令：

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

文件：temperatureInput.json

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

响应:

```
{
    "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."
    }
}
```

注意

- 这些区域有：'sensorId'示例探测器实例不使用它来直接控制或监视传感器。它会自动传递到检测器实例发送的通知中。从那里，它可以用来识别出现故障的传感器（例如，定期发送异常读数的传感器可能即将失效），或者已经离线（当它被用作监视设备心跳的附加探测器模型的输入时）的传感器。这些区域有：'sensorId'如果某个区域的读数经常与平均值不同，也可以帮助识别该区域的温暖或寒冷区域。
- 这些区域有：'areaId'用于将传感器的数据路由到相应的探测器实例。为每个唯一的检测器实例创建一个探测器实例'areaId'在任何消息中收到。请参阅...'key'字段中返回的子位置类型'areaDetectorModel'定义。

探测器模型定义

这些区域有：'areaDetectorModel'示例中有内联注释。

CLI 命令：

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

文件：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"
}
```

响应:

```
{
    "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"
    }
}
```

BatchUpdateDetector示例

您可以使用BatchUpdateDetector将检测器实例置于已知状态的操作，包括计时器和变量值。在以下示例中，BatchUpdateDetector操作会重置受温度监控和控制的区域的运行参数。此操作使您无需删除、重新创建或更新探测器模型即可执行此操作。

CLI 命令：

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

文件：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": [
        ]
      }
    }
  ]
}
```

响应:

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

BatchPutMessage例子

Example 1

使用`BatchPutMessage`操作发送`"seedTemperatureInput"`该消息为受温度控制和监控的给定区域设置运行参数。收到的任何消息Amazon IoT Events它有一个新的`"areaId"`导致创建新的检测器实例。但是新的探测器实例不会将状态更改为`"idle"`然后开始监控温度并控制加热或冷却装置，直到`"seedTemperatureInput"`已收到新区域的消息。

CLI 命令：

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json
```

文件：`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}"
```

```
    }
  ]
}
```

响应:

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

Example

2

使用BatchPutMessage操作发送"temperatureInput"消息用于报告给定控制和监视区域中传感器的温度传感器数据。

CLI 命令：

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json
```

文件：temperatureExample.json

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

响应:

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

Example 3

使用BatchPutMessage操作发送"seedTemperatureInput"用于更改给定区域所需温度值的消息。

CLI 命令：

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json
```

文件：seedSetDesiredTemp.json

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
```

```
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

响应:

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

示例：摄取 MQTT 消息

如果您的传感器计算资源无法使用"BatchPutMessage"API，但可以将其数据发送到Amazon IoT Core消息代理使用轻量级 MQTT 客户端，你可以创建一个Amazon IoT Core用于将消息数据重定向到的主题规则Amazon IoT Events输入。以下是 a 的定义Amazon IoT Events主题规则采用"areaId"和"sensorId"来自MQTT 主题的输入字段，以及"sensorData.temperature"消息负载中的字段"temp"字段，并将这些数据摄入我们的Amazon IoT Events "temperatureInput".

如果您的传感器计算资源无法使用"BatchPutMessage"API，但可以将其数据发送到Amazon IoT Core消息代理使用轻量级 MQTT 客户端，你可以创建一个Amazon IoT Core用于将消息数据重定向到的主题规则Amazon IoT Events输入。以下是 a 的定义Amazon IoT Events主题规则采用"areaId"和"sensorId"来自MQTT 主题的输入字段，以及"sensorData.temperature"消息负载中的字段"temp"字段，并将这些数据摄入我们的Amazon IoT Events "temperatureInput".

CLI 命令：

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

文件：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"
  }
}
```

回应：[无]

如果传感器发送了有关该主题的消息"update/temperature/Area51/03"具有以下有效载荷。

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

这会导致数据被摄入Amazon IoT Events好像以下"BatchPutMessage"API 调用已完成。

```
aws iotevents-data batch-put-message --cli-input-json file://spoofExample.json
```

文件：spoofExample.json

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

示例：生成的Amazon SNS 消息

以下是生成的 SNS 消息的示例"Area51"探测器实例

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

 "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},"eventTriggerD
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature":20.0,"r
{}}},"eventName":"resetHeatCool"}
```

```
Cooling system off command> {"eventTime":1557520274729,"payload":
{"actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192","detector":
{"detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},"eventTriggerD
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature":20.0,"r
{}}},"eventName":"resetHeatCool"}
```

示例： DescribeDetectorAPI

您可以使用DescribeDetector操作可查看探测器实例的当前状态、变量值和计时器。

CLI 命令：

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

响应:

```
{
    "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"
    }
}
```

Amazon IoT Core规则引擎示例

以下规则重新发布Amazon IoT Core作为影子更新请求消息的 MQTT 消息。我们假设这个Amazon IoT Core为由探测器模型控制的每个区域的加热单元和冷却装置定义事物。在此示例中，我们定义了名为的内容`"Area51HeatingUnit"`和`"Area51CoolingUnit"`.

CLI 命令：

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

文件：`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"
        }
      }
    ]
  }
}
```

响应：[空]

CLI 命令：

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

文件：`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"
        }
```

```
      }
    ]
  }
}
```

响应：[空]

CLI 命令：

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

文件：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"
        }
      }
    ]
  }
}
```

响应：[空]

CLI 命令：

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

文件：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"
        }
      }
    ]
  }
}
```

响应：[空]

支持的操作

Amazon IoT Events可以在检测到指定事件或过渡事件时触发操作。您可以定义内置动作以使用计时器或设置变量，或将数据发送到其他计时器Amazon资源。

> Note
>
> 在探测器模型中定义操作时，可以对字符串数据类型的参数使用表达式。有关更多信息，请参阅表达式。

Amazon IoT Events支持以下允许您使用计时器或设置变量的操作：

- `setTimer` (p. 87)创建计时器。
- `resetTimer` (p. 88)重置计时器器。
- `clearTimer` (p. 88)删除计时器器。
- `setVariable` (p. 88)创建一个变量。

Amazon IoT Events支持下列可运行的操作Amazon服务：

- `iotTopicPublish` (p. 89)在 MQTT 主题上发布消息。
- `iotEvents` (p. 90)要将数据发送到Amazon IoT Events作为输入值。
- `iotSiteWise` (p. 90) – 将数据发送到 Amazon IoT SiteWise 中的资产属性。
- `dynamoDB` (p. 92)将数据发送到 Amazon DynamoDB 表。
- `dynamoDBv2` (p. 93)将数据发送到 Amazon DynamoDB 表。
- `firehose` (p. 94)将数据发送到 Amazon Kinesis Data Firehose 流。
- `lambda` (p. 94)调用 AAmazon Lambda函数。
- `sns` (p. 95)以推送通知的形式发送数据。
- `sqs` (p. 95)将数据发送到 Amazon SQS 队列。

使用内置操作

Amazon IoT Events支持以下允许您使用计时器或设置变量的操作：

- `setTimer` (p. 87)创建计时器。
- `resetTimer` (p. 88)重置计时器器。
- `clearTimer` (p. 88)删除计时器器。
- `setVariable` (p. 88)创建一个变量。

设置计时器操作

Set timer action

这些区域有：`setTimer`action 允许你创建一个持续时间以秒为单位的计时器。

More information (2)

当您创建计时器时，必须指定以下参数。

timerName

计时器的名称。

durationExpression

（可选）计时器的持续时间，以秒为单位。

持续戳戳戳向下舍入为最接近的整数。例如，如果将计时器设置为 60.99 秒，则持续时间表达式的计算结果为 60 秒。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 SetTimerAction。

重置计时器操作

Reset timer action

这些区域有：`resetTimer`操作允许您将计时器设置为以前计算的持续戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳

More information (1)

当您重置计时器时，必须指定以下参数。

timerName

计时器的名称。

Amazon IoT Events当您重置计时器时，不会重新计算持续戳戳表达式。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 ResetTimerAction。

清除计时器操作

Clear timer action

这些区域有：`clearTimer`操作允许您删除现有计时器。

More information (1)

当您删除计时器时，必须指定以下参数。

timerName

计时器的名称。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 ClearTimerAction。

设置变量操作

Set variable action

这些区域有：`setVariable`操作允许您创建具有指定值的变量。

More information (2)

当您创建变量时，必须指定以下参数。

variableName

变量的名称。

value

变量的新值。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 SetVariableAction。

使用其他Amazon服务

Amazon IoT Events支持下列可运行的操作Amazon服务：

- iotTopicPublish (p. 89)在 MQTT 主题上发布消息。
- iotEvents (p. 90)要将数据发送到Amazon IoT Events作为输入值。
- iotSiteWise (p. 90) – 将数据发送到 Amazon IoT SiteWise 中的资产属性。
- dynamoDB (p. 92)将数据发送到 Amazon DynamoDB 表。
- dynamoDBv2 (p. 93)将数据发送到 Amazon DynamoDB 表。
- firehose (p. 94)将数据发送到 Amazon Kinesis Data Firehose 流。
- lambda (p. 94)调用 AAmazon Lambda函数。
- sns (p. 95)以推送通知的形式发送数据。
- sqs (p. 95)将数据发送到 Amazon SQS 队列。

Important

- 你必须选择同样的选项Amazon两者均适用的区域Amazon IoT Events还有Amazon要使用的服务。有关受支持的区域的列表，请参阅Amazon IoT Events终端节点和配额在里面Amazon Web Services 一般参考.
- 你必须使用相同的Amazon创建其他区域时的区域Amazon资源Amazon IoT Events操作。如果你切换Amazon区域，您在访问时可能会遇到问题Amazon资源。

默认情况下，Amazon IoT Events 会以 JSON 格式为任何操作生成标准有效负载。此操作负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。要配置操作负载，可以使用内容表达式。有关更多信息，请参阅 。表达式 (p. 97)还有Payload数据类型Amazon IoT EventsAPI 参考.

Amazon IoT Core

IoT topic publish action

这些区域有：iotTopicPublish操作允许您通过发布一条 MQTT 消息Amazon IoT消息代理。有关受支持的区域的列表，请参阅Amazon IoT Core终端节点和配额在里面Amazon Web Services 一般参考.

Amazon IoT 消息代理通过将消息从发布客户端发送到订阅客户端来连接 Amazon IoT 客户端。有关更多信息，请参阅 。消息代理Amazon IoT在里面Amazon IoT开发人员指南.

More information (2)

当您发布 MQTT 消息时，必须指定以下参数。

mqttTopic

接收消息的 MQTT 主题。

您可以使用在检测器模型中创建的变量或输入值在运行时动态定义 MQTT 主题名称。

`payload`

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Note

确保已运行Amazon IoT Events服务角色授予`iot:Publish`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 IotTopicPublishAction。

Amazon IoT Events

IoT Events action

这些区域有：`iotEvents`操作允许您将数据发送到Amazon IoT Events作为输入。有关受支持的区域的列表，请参阅Amazon IoT Events终端节点和配额在里面Amazon Web Services 一般参考.

Amazon IoT Events允许您监控您的设备或设备机群中的故障情况或操作中的更改，并在发生此类事件时触发措施。有关更多信息，请参阅 Amazon IoT Events 开发人员指南中的什么是 Amazon IoT Events？

More information (2)

当您向发送数据时Amazon IoT Events，则必须指定以下参数。

`inputName`

名称Amazon IoT Events接收数据的输入。

`payload`

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Note

确保已运行Amazon IoT Events服务角色授予`iotevents:BatchPutMessage`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 IotEventsAction。

Amazon IoT SiteWise

IoT SiteWise action

这些区域有：`iotSiteWise`action 允许您将数据发送到中的资产属性Amazon IoT SiteWise. 有关受支持的区域的列表，请参阅Amazon IoT SiteWise终端节点和配额在里面Amazon Web Services 一般参考.

Amazon IoT SiteWise 是一种托管服务，可让您轻松地从工业设备中大规模收集、组织和分析数据。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的什么是 Amazon IoT SiteWise？。

More information (11)

当你将数据发送到Amazon IoT SiteWise，则必须指定以下参数。

Important

要接收数据，您必须使用中的现有资产属性Amazon IoT SiteWise.

- 如果您将Amazon IoT Events控制台，您必须指定`propertyAlias`确定目标资产财产。
- 如果您将Amazon CLI，您必须指定`propertyAlias`或者两者都有`assetId`和`propertyId`确定目标资产财产。

有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的将工业数据流映射到资产属性。

propertyAlias

（可选）资产属性的别名。您也可以指定表达式。

assetId

（可选）包含指定属性的资产的 ID。您也可以指定表达式。

propertyId

（可选）资产属性的 ID。您也可以指定表达式。

entryId

（可选）此条目的唯一标识符。您可以使用条目 ID，跟踪在出现故障时哪些数据条目导致了错误。默认值为新的唯一标识符。您也可以指定表达式。

propertyValue

一个包含有关属性值的结构。

quality

（可选）资产属性值的质量。值必须为 `GOOD`、`BAD` 或 `UNCERTAIN`。您也可以指定表达式。

timestamp

（可选）一个包含戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳 如果您未指定此值，则默认值为事件戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳戳

timeInSeconds

采用 Unix 纪元时间格式的时间戳（以秒为单位）。有效范围在 1-31556889864403199 之间。您也可以指定表达式。

offsetInNanos

（可选）从转换时的纳秒偏移量`timeInSeconds`. 有效范围在 0-999999999 之间。您也可以指定表达式。

value

一个包含资产属性值的结构。

Important

您必须指定以下值类型之一，具体取决于指定的资产属性的 `dataType`。有关更多信息，请参阅《Amazon IoT SiteWise API 参考》中的 AssetProperty。

booleanValue

（可选）资产属性值是一个布尔值，该值必须为`TRUE`要么`FALSE`. 您也可以指定表达式。如果使用表达式，则计算结果应为布尔值。

doubleValue

（可选）资产属性值为双精度。您也可以指定表达式。如果使用表达式，则计算结果应为双精度值。

integerValue

（可选）资产属性值是整数。您也可以指定表达式。如果使用表达式，则计算结果应为整数值。

stringValue

（可选）资产属性值是字符串。您也可以指定表达式。如果使用表达式，则计算结果应为字符串值。

Note

确保已运行Amazon IoT Events服务角色授予`iotsitewise:BatchPutAssetPropertyValue`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 IotSiteWiseAction。

Amazon DynamoDB

DynamoDB action

这些区域有：`dynamoDB`操作允许您将数据发送到 DynamoDB 表。DynamoDB 表中有一个列用于接收指定操作负载中的所有属性/值对。有关受支持的区域的列表，请参阅Amazon DynamoDB 终端节点和配额在里面Amazon Web Services 一般参考.

Amazon DynamoDB 是一种全托管 NoSQL 数据库服务，提供快速而可预测的性能，能够实现无缝扩展。有关更多信息，请参阅 。什么是 DynamoDB？在里面Amazon DynamoDB D 开发人员指南.

More information (10)

将数据发送到 DynamoDB 表的一列时，必须指定以下参数。

tableName

接收数据的 DynamoDB 表的名称。这些区域有：`tableName`值必须与 DynamoDB 表的表名匹配。您也可以指定表达式。

hashKeyField

哈希键（也称为分区键）的名称。这些区域有：`hashKeyField`值必须与 DynamoDB 表的分区键匹配。您也可以指定表达式。

hashKeyType

（可选）哈希键的数据类型。哈希键类型的值必须为`STRING`要么`NUMBER`. 默认为 `STRING`。您也可以指定表达式。

hashKeyValue

哈希键的值。这些区域有：`hashKeyValue`使用替换模板。这些模板在运行时提供数据。您也可以指定表达式。

rangeKeyField

（可选）范围键（也称为排序键）的名称。这些区域有：`rangeKeyField`值必须与 DynamoDB 表的排序键匹配。您也可以指定表达式。

rangeKeyType

（可选）范围键的数据类型。哈希键类型的值必须为STRING要么NUMBER. 默认为 STRING。您也可以指定表达式。

rangeKeyValue

（可选）范围键的值。这些区域有：rangeKeyValue使用替换模板。这些模板在运行时提供数据。您也可以指定表达式。

operation

（可选）要执行的操作的类型。您也可以指定表达式。操作值必须为以下数值之一：

- INSERT - 将数据作为新项插入到 DynamoDB 表中。这是默认值。
- UPDATE - 使用新数据更新 DynamoDB 表的现有项。
- DELETE-从 DynamoDB 表中删除现有项。

payloadField

（可选）接收操作负载的 DynamoDB 列的名称。默认名称为 payload。您也可以指定表达式。

payload

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

如果指定的负载类型是字符串，DynamoDBAction将非 JSON 数据作为二进制数据发送到 DynamoDB 表。DynamoDB 控制台以 Base64 编码文本格式显示数据。payloadField 值为 *payload-field*_raw。您也可以指定表达式。

Note

确保已运行Amazon IoT Events服务角色授予dynamodb:PutItem权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅 。DynamoDBAction在里面Amazon IoT EventsAPI 参考.

Amazon DynamoDB (AAM)

DynamoDBv2 action

这些区域有：dynamoDBv2操作允许您将数据写入 DynamoDB 表。DynamoDB 表中有一个单独的列用于接收指定操作负载中的一个属性/值对。有关受支持的区域的列表，请参阅Amazon DynamoDB 终端节点和配额在里面Amazon Web Services 一般参考.

Amazon DynamoDB 是一种全托管 NoSQL 数据库服务，提供快速而可预测的性能，能够实现无缝扩展。有关更多信息，请参阅 。什么是 DynamoDB？在里面Amazon DynamoDB.

More information (2)

将数据发送到 DynamoDB 表的多列时，必须指定以下参数。

tableName

接收数据的 DynamoDB 表的名称。您也可以指定表达式。

payload

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Important

负载类型必须为 JSON。您也可以指定表达式。

Note

确保已运行Amazon IoT Events服务角色授予`dynamodb:PutItem`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅 。DynamoDBv2Action在里面Amazon IoT EventsAPI 参考.

Amazon Kinesis Data Firehose

Firehose action

这些区域有：`firehose`操作允许您将数据发送到 Kinesis Data Firehose 传输流。有关受支持的区域的列表，请参阅Amazon Kinesis Data Firehose 终端节点和配额在里面Amazon Web Services 一般参考.

Amazon Kinesis Data Firehose 是一项完全托管的服务，用于向Amazon Simple Storage（Amazon Simple Storage）提供实时流数据的服务，用于向Amazon Simple S OpenSearch 服务（服务（服务OpenSearch 服务）和 Splunk。有关更多信息，请参阅 。Amazon Kinesis Data Firehose在里面Amazon Kinesis Data Firehose.

More information (3)

将数据发送到 Kinesis Data Firehose 传输流时，必须指定以下参数。

`deliveryStreamName`

接收数据的 Kinesis Data Firehose 传输流的名称。

`separator`

（可选）您可以使用字符分隔发送到 Kinesis Data Firehose 传输流的连续数据。分隔值必须是`'\n'`（换行符），`'\t'`（选项卡），`'\r\n'`（Windows 换行），或`','`（逗号）。

`payload`

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Note

确保已运行Amazon IoT Events服务角色授予`firehose:PutRecord`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 FirehoseAction。

Amazon Lambda

Lambda action

这些区域有：`lambda`操作允许您调用 Lambda 函数。有关受支持的区域的列表，请参阅Amazon Lambda终端节点和配额在里面Amazon Web Services 一般参考.

Amazon Lambda 是一项计算服务，可使您无需预配置或管理服务器即可运行代码。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的什么是 Amazon Lambda？

More information (2)

当您调用 Lambda 函数时，必须指定以下参数。

`functionArn`

要调用的 Lambda 函数的 ARN。

`payload`

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Note

确保已运行Amazon IoT Events服务角色授予`lambda:InvokeFunction`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅《Amazon IoT Events API 参考》中的 LambdaAction。

Amazon Simple Notification Service

SNS action

这些区域有：`sns`主题发布操作允许您发布 Amazon SNS 消息。有关受支持的区域的列表，请参阅Amazon Simple Notification Service 终端节点和配额在里面Amazon Web Services 一般参考.

Amazon Simple Notification（Amazon Simple Notification）是一项 Web 服务，用于协调和管理向订阅终端节点或客户端交付或发送消息的过程。有关更多信息，请参阅 。什么是 Amazon SNS？在里面Amazon Simple Notifice.

More information (2)

当您发布 Amazon SNS 消息时，必须指定以下参数。

`targetArn`

接收消息的 Amazon SNS 目标的 ARN。

`payload`

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Note

确保已运行Amazon IoT Events服务角色授予`sns:Publish`权限。有关更多信息，请参阅 适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅 。SNS NSTopicPublishAction在里面Amazon IoT EventsAPI 参考.

Amazon Simple Queue Service

SQS action

这些区域有：`sns`操作允许您将数据发送到 Amazon SQS 队列。有关受支持的区域的列表，请参阅Amazon Simple Queue Service 终端节点和配额在里面Amazon Web Services 一般参考.

Amazon Simple Que Que (Amazon SQS) 提供一个安全、持久且可用的托管队列，允许您集成和分离分布式软件系统和组件。有关更多信息，请参阅 。Amazon Simple Queueue在里面Amazon Simple Queue.

More information (3)

当您向 Amazon SQS 队列发送数据时，必须指定以下参数。

`queueUrl`

接收数据的Amazon SQS 队列的 URL。

`useBase64`

（可选）Amazon IoT Events将数据编码为 Base64 文本（如果指定）`TRUE`. 默认为 `FALSE`。

`payload`

（可选）默认负载包含有关检测器模型实例和触发操作的事件的信息的所有属性/值对。此外，您还可以自定义负载。有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.

Note

确保已运行Amazon IoT Events服务角色授予`sqs:SendMessage`权限。有关更多信息，请参阅适用于 Amazon IoT Events 的 Identity and Access Management (p. 157)。

有关更多信息，请参阅 。SNS NSTopicPublishAction在里面Amazon IoT EventsAPI 参考.

您还可以使用 Amazon SNS 和Amazon IoT Core触发规则引擎Amazon Lambda函数。这使得使用其他服务（例如 Amazon Connect，甚至是公司企业资源规划 (ERP) 应用程序）采取行动成为可能。

Note

要实时收集和处理大型数据流记录，您可以使用其他Amazon服务，例如Amazon Kinesis. 然后，您可以完成初步分析，然后将结果发送至Amazon IoT Events作为探测器的输入。

表达式

Amazon IoT Events在创建和更新探测器模型时提供了多种指定值的方法。您可以使用表达式来指定文字值，或者Amazon IoT Events可以在指定特定值之前对表达式求值。

语法

您可以使用文字、运算符、函数、引用和替代模板Amazon IoT Events表达式。

文本

- 整数
- 小数
- 字符串
- 布尔值

运算符

一元结果

- 不是（布尔值）：`!`
- 不是（按位）：`~`
- 减号（算术）：`-`

字符串

- 连接结果：`+`

  两个操作数都必须是字符串。字符串文本必须括在单引号 (') 内。

  例如：`'my' + 'string'->'mystring'`

算术

- 另外：`+`

  两个操作数都必须是数字。

- 减：`-`
- 除：`/`

  除法结果是四舍五入的整数值，除非至少有一个操作数（除数或除数）是十进制值。

- 乘法：`*`

按位（整数）

- 或：`|`

  例如：`13 | 5->13`

- 和：`&`

  例如：`13 & 5->5`

- 异或：**^**

  例如：`13 ^ 5->8`
- 不是：**~**

  例如：`~13->-14`

布尔值

- 小于：**<**
- 小于或等于：**<=**
- 等于：**==**
- 不等于：**!=**
- 大于或等于：**>=**
- 大于：**>**
- 和：**&&**
- 或：**||**

  Note

  当子表达式为`||`包含未定义的数据，该子表达式被视为`false`.

圆括号

您可以使用圆括号对表达式中的术语进行分组。

函数

内置函数

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

求解`true`如果指定的计时器已过。替换"*`#####`*"使用您定义的计时器的名称，用引号括起来。在事件操作中，您可以定义一个计时器，然后启动计时器、重置计时器或清除先前定义的计时器。查看背景`detectorModelDefinition.states.onInput|onEnter|onExit.events.actions.setTimer.timerName`.

在一个状态下设置的计时器可以在不同的状态下被引用。在进入引用计时器的状态之前，必须访问创建计时器的状态。

例如，探测器模型有两种状态， TemperatureChecked 和 RecordUpdated. 您在中创建了一个计时器 TemperatureChecked 状态。你必须访问 TemperatureChecked 在使用计时器之前，请先注明 RecordUpdated 状态。

为了确保准确性，计时器设置的最短时间为 60 秒。

Note

`timeout()`回报`true`仅在实际计时器到期后第一次检查并返回`false`此后。

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

计算结果为转换为指定类型的表达式的值。这些区域有：*`##`*值必须是`String`,`Boolean`，或`Decimal`. 使用这些关键字之一或计算结果为包含该关键字的字符串的表达式。只有以下转换成功并返回有效值：

- 布尔值

  返回字符串`"true"`要么`"false"`.
- DECIMAL

- 字符串
- 字符串

  指定的字符串必须是十进制数的有效表示形式，或convert()失败。

如果convert()不返回有效值，它所属的表达式也是无效的。此结果等同于false而且不会触发actions或者过渡到nextState指定为表达式发生的事件的一部分。

isNull(*expression*)

求解true如果表达式返回 null。例如，如果输入MyInput接收消息{ "a": null }，则以下计算结果为true，但是isUndefined($input.MyInput.a)求解false.

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

isUndefined(*expression*)

求解true如果表达式未定义。例如，如果输入MyInput接收消息{ "a": null }，则以下计算结果为false，但是isNull($input.MyInput.a)求解true.

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

triggerType("*type*")

这些区域有：*##*值可以是"Message"要么"Timer". 求解true如果由于计时器已过期而对其出现的事件条件进行评估，如下例所示。

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

或者收到了输入消息。

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

currentInput("*input*")

求解true如果因为收到指定的输入消息而正在评估其出现的事件条件。例如，如果输入Command接收消息{ "value": "Abort" }，则以下计算结果为true.

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

使用此函数验证是否因为已收到特定输入而计时器尚未过期而正在评估条件，如以下表达式所示。

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

字符串匹配函数

startsWith(*expression1*, *expression2*)

求解true如果第一个字符串表达式以第二个字符串表达式开头。例如，如果输入MyInput接收消息{ "status": "offline"}，则以下计算结果为true.

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

这两个表达式的计算结果必须为字符串值。如果任一表达式的计算结果不为字符串值，则该函数的结果未定义。不进行任何转换。

endsWith(*expression1*, *expression2*)

求解true如果第一个字符串表达式以第二个字符串表达式结尾。例如，如果输入MyInput接收消息{ "status": "offline" }，则以下计算结果为true.

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

这两个表达式的计算结果必须为字符串值。如果任一表达式的计算结果不为字符串值，则该函数的结果未定义。不进行任何转换。

contains(*expression1*, *expression2*)

求解true如果第一个字符串表达式包含第二个字符串表达式。例如，如果输入MyInput接收消息{ "status": "offline" }，则以下计算结果为true.

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

这两个表达式的计算结果必须为字符串值。如果任一表达式的计算结果不为字符串值，则该函数的结果未定义。不进行任何转换。

按位整数操作函数

bitor(*expression1*, *expression2*)

计算整数表达式的按位 OR（二进制 OR 运算对整数的相应位执行）。例如，如果输入MyInput接收消息{ "value1": 13, "value2": 5 }，则以下计算结果为13.

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

两个表达式的计算结果都必须为整数值。如果任一表达式的计算结果不为整数值，则该函数的结果未定义。不进行任何转换。

bitand(*expression1*, *expression2*)

计算整数表达式的按位 AND（二进制 AND 运算对整数的相应位执行）。例如，如果输入MyInput接收消息{ "value1": 13, "value2": 5 }，则以下计算结果为5.

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

两个表达式的计算结果都必须为整数值。如果任一表达式的计算结果不为整数值，则该函数的结果未定义。不进行任何转换。

bitxor(*expression1*, *expression2*)

计算整数表达式的按位 XOR（二进制 XOR 运算对整数的相应位执行）。例如，如果输入MyInput接收消息{ "value1": 13, "value2": 5 }，则以下计算结果为8.

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

两个表达式的计算结果都必须为整数值。如果任一表达式的计算结果不为整数值，则该函数的结果未定义。不进行任何转换。

bitnot(*expression*)

计算整数表达式的按位 NOT（二进制 NOT 运算对整数的位执行）。例如，如果输入MyInput接收消息{ "value": 13 }，则以下计算结果为-14.

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

两个表达式的计算结果都必须为整数值。如果任一表达式的计算结果不为整数值，则该函数的结果未定义。不进行任何转换。

参考

输入

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

`input-name`是您使用创建的输入CreateInput操作。

例如，如果您具有名为`TemperatureInput`你为此定义的`inputDefinition.attributes.jsonPath`条目，这些值可能会出现在以下可用字段中。

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

要引用`temperature`字段中，使用以下命令。

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

对于值为数组的字段，您可以使用以下方法引用数组的成员`[n]`. 例如，给定以下值：

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

值`78.8`可以使用以下命令进行引用。

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

Variables

`$variable.variable-name`

这些区域有：*variable-name*是您使用定义的变量CreateDetectorModel操作。

例如，如果你有一个名为的变量`TechnicianID`你用它定义的`detectorDefinition.states.onInputEvents.actions.setVariable.variableName`，你可以使用以下命令引用最近给变量的（字符串）值。

```
$variable.TechnicianID
```

您只能使用以下方式设置变量的值`setVariable`操作。您不能为表达式中的变量赋值。变量不能被取消设置。例如，您不能为其分配值`null`.

Note

在使用不遵循（正则表达式）模式的标识符的引用中`[a-zA-Z][a-zA-Z0-9_]*`，你必须用反引号将这些标识符括起来 (`)。例如，对名为的输入的引用`MyInput`用一个名为的字段`_value`必须将此字段指定为`$input.MyInput.`_value`.

在表达式中使用引用时，请检查以下几点：

- 当您使用引用作为具有一个或多个运算符的操作数时，请确保您引用的所有数据类型都兼容。

  例如，在以下表达式中，INTEGER2是两个的操作数==和&&运算符。为了确保操作数兼容，`$variable.testVariable + 1`和`$variable.testVariable`必须引用整数或小数。

  此外，INTEGER1是的操作数+操作符。因此，`$variable.testVariable`必须引用整数或小数。

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

- 当您使用引用作为传递给函数的参数时，请确保该函数支持您引用的数据类型。

  例如以下几点`timeout("time-name")`函数需要一个以双引号作为参数的字符串。如果你使用参考来表示`#####`值，则必须引用带双引号的字符串。

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

  Note

  对于`convert(type, expression)`函数，如果你使用引用来表示`##`值，您的参考文献的评估结果必须为`String`,`Decimal`，或`Boolean`.

Amazon IoT Events表达式支持整数、十进制、字符串和布尔数据类型。下表提供了不兼容类型对的列表。

不兼容的类型对
整数、字符串
整数，布尔值
十进数，字符串
十进数，布尔值
字符串

替换模板

`'${expression}'`

这些区域有：`${}`将该字符串标识为插值字符串。这些区域有：`expression`可以是任意Amazon IoT Events表达式。这包括运算符、函数和引用。

例如，您使用了SetVariableAction定义变量的操作。`variableName` 是 `SensorID`，`value` 是 `10`。您可以创建以下替代模板。

替换模板	结果字符串
`'${'Sensor ' + $variable.SensorID}'`	`"Sensor 10"`
`'Sensor ' + '${$variable.SensorID + 1}'`	`"Sensor 11"`

替换模板	结果字符串
`'Sensor 10: ${$variable.SensorID == 10}'`	`"Sensor 10: true"`
`'{\"sensor\":\"${$variable.SensorID + 1}\"}'`	`"{\"sensor\":\"11\"}"`
`'{\"sensor\":${$variable.SensorID + 1}}'`	`"{\"sensor\":11}"`

表达式用法

您可以通过以下方式指定探测器模型中的值：

- 在中输入支持的表达式Amazon IoT Events控制台。
- 将表达式传递给Amazon IoT EventsAPI 作为参数。

表达式支持文字、运算符、函数、引用和替代模板。

> **Important**
>
> 您的表达式必须引用整数、十进制、字符串或布尔值。

ritingAmazon IoT Events表达式

请参阅以下示例，以帮助您撰写您的Amazon IoT Events表达式：

文本

对于文字值，表达式必须包含单引号。布尔值必须是`true`要么`false`.

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

参考

对于引用，必须指定变量或输入值。

- 以下输入引用十进制数，`10.01`.

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

- 以下变量引用一个字符串，`Greenhouse Temperature Table`.

```
$variable.TableName
```

替换模板

对于替代模板，您必须使用 `${}`，且模板必须在单引号内。替代模板还可以包含文字、运算符、函数、引用和替代模板的组合。

- 以下表达式的计算结果是一个字符串，`50.018 in Fahrenheit`.

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

- 以下表达式的计算结果是一个字符串，{\"sensor_id\":\"Sensor_1\",\"temperature\": \"50.018\"}.

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

字符串连接

对于字符串串联，必须使用 +。字符串串联还可以包含文字、运算符、函数、引用和替代模板的组合。

- 以下表达式的计算结果是一个字符串，Greenhouse Temperature Table 2000-01-01.

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

探测器模型示例

本节包含探测器模型和输入的示例。

主题

暖通空调温度控制

后台故事

此示例实现了具有以下功能的温度控制模型（恒温器）：

- 您定义的一个探测器模型可以监视和控制多个区域。（将为每个区域创建一个探测器实例。）
- 每个探测器实例从放置在每个控制区域的多个传感器接收温度数据。
- 您可以随时更改每个区域的所需温度（设定点）。
- 您可以为每个区域定义操作参数并随时更改这些参数。
- 您可以随时向某个区域添加传感器或从该区域删除传感器。
- 您可以启用加热和冷却装置的最短运行时间，以保护它们免受损坏。
- 探测器将拒绝并报告异常的传感器读数。
- 您可以定义紧急温度设定点。如果任何一个传感器报告的温度高于或低于您定义的设定点，则加热或冷却装置将立即启动，探测器将报告该温度峰值。

该示例演示以下功能功能：

- 创建事件探测器模型。
- 创建输入。
- 将输入引入探测器模型。
- 评估触发条件。
- 引用条件中的状态变量并根据条件设置变量的值。
- 参考条件中的计时器，并根据条件设置计时器。
- 采取发送Amazon SNS 和 MQTT 消息的操作。

输入定义

一个`"seedTemperatureInput"`用于为某个区域创建探测器实例并定义其运行参数。

使用CCI 命令：

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

文件：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" }
    ]
  }
}
```

响应:

```
{
    "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."
    }
}
```

一个"temperatureInput"应根据需要由每个区域的每个传感器发送。

使用CCI 命令：

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

文件：temperatureInput.json

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

响应:

```
{
    "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."
    }
}
```

探测器模型定义

这些区域有："areaDetectorModel"定义每个探测器实例的工作原理。每个"state machine"实例将摄取温度传感器读数，然后根据这些读数更改状态并发送控制消息。

使用CCI 命令：

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

文件：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"
}
```

响应:

```
{
    "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"
    }
}
```

BatchUpdateDetector 示例

在此示例中，"BatchUpdateDetector"用于更改正在运行的探测器实例的操作参数。

使用CCI 命令：

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

文件：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": [
        ]
      }
    }
  ]
}
```

响应:

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

BatchPutMessage 例子

在此示例中，"BatchPutMessage"用于为某个区域创建探测器实例并定义初始操作参数。

使用CCI 命令：

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json
```

文件：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}"
    }
  ]
}
```

响应:

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

在此示例中，"BatchPutMessage"用于报告某个区域中单个传感器的温度传感器读数。

使用CCI 命令：

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json
```

文件：temperatureExample.json

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

响应:

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

在此示例中，"BatchPutMessage"用于改变某个区域所需的温度。

使用CCI 命令：

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json
```

文件：seedSetDesiredTemp.json

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

响应:

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

由生成的 Amazon SNS 消息示例Area51探测器实例：

```
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"
}
```

在此示例中，我们使用的是"DescribeDetector"用于获取有关检测器实例当前状态信息的 API。

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

响应:


```
{
    "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"
    }
}
```

Amazon IoT Core规则引擎示例

以下规则重新发布Amazon IoT Events作为影子更新请求消息的 MQTT 消息。我们假设是Amazon IoT Core为由探测器模型控制的每个区域的加热单元和冷却装置定义事物。

在此示例中，我们定义了名为的事物`"Area51HeatingUnit"`和`"Area51CoolingUnit"`.

使用CCI 命令：

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

文件：`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"
        }
      }
    ]
  }
}
```

响应：[空]

使用CCI 命令：

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

文件：`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"
        }
      }
    ]
  }
}
```

响应：[空]

使用CCI 命令：

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

文件：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"
        }
      }
    ]
  }
}
```

响应：[空]

使用CCI 命令：

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

文件：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"
        }
      }
    ]
  }
}
```

响应：[空]

起重机

后台故事

许多起重机的操作员都想检测机器何时需要维护或更换，并触发相应的通知。每台起重机都有一个马达。电动机发出包含压力和温度信息的消息（输入）。操作员需要两个级别的事件探测器：

- 起重机级事件探测器
- 马达级事件探测器

使用来自马达的消息（其中包含元数据，两者都是`"craneId"`还有`"motorId"`），操作员可以使用适当的路由执行两个级别的事件探测器。当事件条件得到满足时，应向相应的 Amazon SNS 主题发送通知。操作员可以配置探测器模型，这样就不会发出重复的通知。

该示例演示以下功能功能：

- 创建、读取、更新、删除 (CRUD) 输入文件。
- 创建、读取、删除 (CRUD) 事件检测器模型和不同版本的事件探测器。
- 将一个输入路由到多个事件探测器。
- 将输入引入探测器模型。
- 评估触发条件和生命周期事件。
- 能够在条件中引用状态变量并根据条件设置其值。
- 使用定义、状态、触发器评估器和操作执行器进行运行时编排。
- 在中执行操作`ActionsExecutor`使用 SNS 目标。

命令

```
#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

#Send HighTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highTemperatureMessage.json

#Send LowPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowPressureMessage.json

#Send LowTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowTemperatureMessage.json
```

探测器模型数

文件：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"
}
```

更新现有探测器模型。文件：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"
```

```
}
```

文件：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"
}
```

更新现有探测器模型。文件：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"
}
```

输入

文件：pressureInput.json

```
{
```

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

文件：temperatureInput.json

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

消息

文件：highPressureMessage.json

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

        }
    ]
}
```

文件：highTemperatureMessage.json

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

文件：lowPressureMessage.json

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

```
        }
    ]
}
```

文件：lowTemperatureMessage.json

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

使用传感器和应用程序进行事件检测

此探测器模型是可用的模板之一Amazon IoT Events控制台。为方便起见，我们还将对此进行了介绍。

```
{
    "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"
    }
}
```

设备 HeartBeat

此探测器模型是可用的模板之一Amazon IoT Events控制台。为方便起见，我们还将对此进行了介绍。

```
{
    "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 警报

此探测器模型是可用的模板之一Amazon IoT Events控制台。为方便起见，我们还将对此进行了介绍。

```
{
    "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"
}
```

简单告警

此探测器模型是可用的模板之一Amazon IoT Events控制台。为方便起见，我们还将对此进行了介绍。

```
{
    "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"
}
```

使用警报进行监控

Amazon IoT Events警报可帮助您监控数据的变化。这些数据可以是您衡量设备和过程的指标。您可以创建警报，在超过阈值时发送通知。警报可帮助您发现问题、简化维护并优化设备和流程的性能。

警报是警报模型的实例。警报模型指定要检测的内容、何时发送通知、谁收到通知等等。您也可以指定一个或多个支持的操作警报状态改变时发生的情况。Amazon IoT Events路线输入属性从您的数据中派生到相应的警报。如果您监控的数据超出指定范围，则会调用警报。您也可以确认闹钟或将其设置为暂停模式。

> Note
>
> 警报通知功能在中国（北京）区域不可用。

使用 Amazon IoT SiteWise

您可以使用Amazon IoT Events警报以监控Amazon IoT SiteWise.Amazon IoT SiteWise将资产属性值发送到Amazon IoT Events警报。Amazon IoT Events将警报状态发送到Amazon IoT SiteWise.

Amazon IoT SiteWise还支持外部警报。如果您使用外部警报，则可以选择外部警报Amazon IoT SiteWise并有一个返回警报状态数据的解决方案。外部警报包含一个测量属性，用于提取警报状态数据。

Amazon IoT SiteWise不评估外部警报的状态。此外，当警报状态发生变化时，您无法确认或暂停外部警报。

您可以使用 SiteWise 监控功能可查看外部警报的状态 SiteWise 监控门户。

有关更多信息，请参阅 。使用警报进行监控在Amazon IoT SiteWise用户指南和使用警报进行监控在SiteWise 监视器应用指南.

确认流程

创建警报模型时，您可以选择是否启用确认流。如果您启用确认流，则警报状态发生变化时，您的团队会收到通知。您的团队可以确认警报并留言。例如，您可以包括警报信息以及为解决问题将要采取的措施。如果您监控的数据超出指定范围，则会调用警报。

警报具有以下状态：

`DISABLED`

当警报进入时`DISABLED`状态，它还没准备好评估数据。要启用警报，必须将警报更改为`NORMAL`状态。

`NORMAL`

当警报进入时`NORMAL`状态，已经准备好评估数据了。

`ACTIVE`

如果警报在`ACTIVE`状态，警报被调用。您正在监控的数据超出指定范围。

`ACKNOWLEDGED`

当警报进入时`ACKNOWLEDGED`状态，警报已调用，您已确认警报。

`LATCHED`

警报已调用，但您在一段时间后没有确认警报。警报会自动更改为`NORMAL`状态。

`SNOOZE_DISABLED`

当警报进入时`SNOOZE_DISABLED`状态，警报在指定的时间段内处于禁用状态。贪睡时间过后，闹钟会自动变为`NORMAL`状态。

创建警报模型

您可以使用Amazon IoT Events警报用于监控您的数据，并在超过阈值时收到通知。警报提供用于创建或配置警报模型的参数。您可以使用Amazon IoT Events控制台或Amazon IoT Events用于创建或配置警报模型的API。配置警报模型时，更改将在新数据到达时生效。

要求

创建警报模型时存在以下要求。

- 您可以创建警报模型来监控中的输入属性Amazon IoT Events或资产财产Amazon IoT SiteWise.
  - 如果您选择监控中的输入属性Amazon IoT Events，在创建警报模型之前：
    - 步骤 1: 阅读中的概述创建输入.
    - 步骤 2: 请阅读以下说明在导航窗格中创建输入.
  - 如果您选择监控资产财产，则必须创建资产模型在Amazon IoT SiteWise在创建警报模型之前。
- 您必须拥有允许警报执行操作和访问的 IAM 角色Amazon资源。有关更多信息，请参阅 。为设置权限Amazon IoT Events.
- 所有的Amazon本教程使用的资源必须相同Amazon区域。

创建警报模型（控制台）

下面的介绍了如何创建警报模型来监控Amazon IoT Events属性在Amazon IoT Events控制台。

1. 登录到 Amazon IoT Events 控制台。
2. 在导航窗格中，选择警报模型.
3. 在警报模型页面，选择创建警报模型.
4. 在警报型号详情部分，执行以下操作：
   a. 输入唯一名称。
   b. （可选）输入描述。
5. 在警报目标部分，执行以下操作：

   Important

   如果选择...Amazon IoT SiteWise资产财产，你必须已经在中创建了资产模型Amazon IoT SiteWise.

   a. 选择Amazon IoT Events输入属性.
   b. 选择输入。
   c. 选择输入属性键。此输入属性用作创建警报的密钥。Amazon IoT Events将与此密钥相关的输入路由到警报。
6. 在阈值定义部分，您可以定义输入属性、阈值和比较运算符Amazon IoT Events用于更改警报的状态。
   a. 对于输入属性请选择要监控的属性。

      每次该输入属性接收到新数据时，都会对其进行评估以确定警报的状态。
   b. 对于操作符，选择比较运算符。该运算符将您的输入属性与您的属性的阈值进行比较。

      可从以下选项中进行选择：

      - > 大于

- >= 大于或等于
- < 小于
- <= 小于或等于
- = 等于
- ! = 不等于

c. 用于阈值值，在中输入数字或选择一个属性Amazon IoT Events输入。Amazon IoT Events将此值与您选择的输入属性的值进行比较。

d. （可选）For严重性，使用您的团队理解的数字来反映此警报的严重性。

7. （可选）在通知设置部分，配置警报的通知设置。

   Note

   警报通知功能在中国（北京）区域不可用。

   您最多可以添加 10 条通知。对于通知 1，执行以下操作：

   a. 对于协议，请从以下选项中进行选择：

      - 电子邮件和短信-警报发送短信通知和电子邮件通知。
      - Email（电子邮件）-警报会发送电子邮件通知。
      - 文本-警报发送短信通知。

   b. 对于Sender，指定可以发送有关此警报的通知的电子邮件地址。

      要向发件人列表添加更多电子邮件地址，请选择添加发件人.

   c. （可选）For收件人，选择收件人。

      要向您的收件人列表添加更多用户，请选择添加新用户. 您必须将新用户添加到您的Amazon IAM Identity Center (successor to Amazon Single Sign-On) 目录，然后才能将它们添加到警报模型中。有关更多信息，请参阅 管理收件人 (p. 156)。

   d. （可选）For其他自定义消息，输入一条消息，描述警报检测到的内容以及收件人应采取的操作。

8. 在实例部分，您可以启用或禁用基于此警报模型创建的所有警报实例。

9. 在高级设置部分，执行以下操作：

   a. 对于确认流程，您可以启用或禁用通知。

      - 如果选择...Enabled，当警报状态发生变化时，您会收到通知。在警报状态恢复正常之前，您必须确认通知。
      - 如果选择...Disabled，无需执行任何操作。当测量值返回到指定范围时，警报会自动更改为正常状态。

      有关更多信息，请参阅 确认流程 (p. 146)。

   b. 对于Permissions (权限)，请选择以下任一选项：

      - 您可以从创建一个新角色Amazonpolicy和Amazon IoT Events会自动为您创建 IAM 角色。
      - 您可以使用现有 IAM 角色允许此警报模型执行操作并访问其他Amazon资源。

      有关更多信息，请参阅 。适用于 的 Identity and Access ManagementAmazon IoT Events.

   c. 对于其他通知设置，你可以编辑你的Amazon Lambda管理警报通知的功能。请选择：Amazon Lambda函数:

      - 创建新的Amazon Lambda功能-Amazon IoT Events创建新的Amazon Lambda为你服务。
      - 使用现有的Amazon Lambda功能-使用现有Amazon Lambda通过选择一个函数Amazon Lambda函数名称。

有关可能操作的更多信息，请参阅。使用其他Amazon服务 (p. 89).

d. （可选）For设置状态操作，可以添加一个或多个Amazon IoT Events警报状态发生变化时要采取的操作。

10. （可选）可以添加标签管理您的警报。有关更多信息，请参阅 。给您加标签Amazon IoT Events资源.
11. 选择Create（创建）。

响应警报

如果你启用了确认流程，当警报状态发生变化时，您会收到通知。要响应警报，您可以确认、禁用、启用、重置或暂停警报。

响应警报（控制台）

下面的介绍了如何对警报进行响应Amazon IoT Events控制台。

1. 登录到 Amazon IoT Events 控制台。
2. 在导航窗格中，选择警报模型.
3. 选择目标警报模型。
4. 在警报的列表部分，选择目标警报。
5. 您可以选择：操作：

   - 承认-警报变为ACKNOWLEDGED状态。
   - 禁用-警报变为DISABLED状态。
   - 启用-警报变为NORMAL状态。
   - 重置-警报变为NORMAL状态。
   - 打瞌睡，然后执行以下操作：
     1. 选择暂停长度或者输入集群自定义贪睡长度.
     2. 选择Save（保存）。

     警报变为SNOOZE_DISABLEDstate

   有关警报状态的更多信息，请参阅。确认流程 (p. 146).

响应警报 (API)

要响应一个或多个警报，您可以使用以下方法Amazon IoT EventsAPI 操作：

- BatchAcknowledgeAlarm
- BatchDisableAlarm
- BatchEnableAlarm
- BatchResetAlarm
- BatchSnoozeAlarm

管理警报通知

Amazon IoT Events使用 Lambda 函数来管理警报通知。您可以使用提供的 Lambda 函数Amazon IoT Events或者创建一个新的。

Note

警报通知功能在中国（北京）区域不可用。

创建 Lambda 函数

Amazon IoT Events提供了 Lambda 函数，使警报能够发送和接收电子邮件和短信通知。

要求

在为警报创建 Lambda 函数时，请选择

- 如果您的警报发送电子邮件或短信通知，则您必须拥有允许的 IAM 角色Amazon Lambda与Amazon SES和Amazon SNS 合作。

  示例策略：

```
{
    "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:*:*:*"
        }
    ]
}
```

- 你必须选择相同的Amazon两者均适用的区域Amazon IoT Events和Amazon Lambda. 有关受支持的区域的列表，请参阅。Amazon IoT Events终端节点和配额和Amazon Lambda终端节点和配额在Amazon Web Services 一般参考.

部署 Lambda 函数

本教程使用的是Amazon CloudFormation部署 Lambda 函数的模板。此模板会自动创建一个 IAM 角色，允许 Lambda 函数与 Amazon SES 和 Amazon SNS 配合使用。

下面的介绍了如何使用Amazon Command Line Interface(Amazon CLI）创建集群 CloudFormation 堆栈。

1. 在设备的终端中，运行`aws --version`来检查你是否安装了Amazon CLI. 有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装 Amazon CLI。
2. Run（运行）`aws configure list`检查您是否配置了Amazon CLI在Amazon拥有你所有东西的区域Amazon本教程的资源。有关更多信息，请参阅 。配置Amazon CLI在Amazon Command Line Interface 用户指南
3. 下载 CloudFormation 模板，通知 lambda.template.yaml.zip.

   Note

   如果您在下载文件时遇到困难，也可以在CloudFormation 模板 (p. 153).
4. 解压缩内容并将其作为 `notificationLambda.template.yaml` 保存在本地。
5. 在您的设备上打开终端并将当前目录导航到您下载的目录`notificationLambda.template.yaml`file。
6. 创建集群 CloudFormation stack，运行以下命令：

```
aws cloudformation create-stack --stack-name notificationLambda-stack --template-body
 file://notificationLambda.template.yaml --capabilities CAPABILITY_IAM
```

你可以修改这个 CloudFormation 用于自定义 Lambda 函数及其行为的模板。

Note

Amazon Lambda会针对函数错误重试两次。如果该函数没有足够的容量来处理所有传入请求，则事件可能会在队列中等待数小时或数天才能发送到该函数。您可以在函数上配置未传送消息队列 (DLQ) 以捕获未成功处理的事件。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的异步调用。

您也可以在中创建或配置堆栈 CloudFormation 控制台。有关更多信息，请参阅 。使用堆栈，在Amazon CloudFormation用户指南.

创建自定义 Lambda 函数

您可以创建 Lambda 函数或修改提供的 Lambda 函数Amazon IoT Events.

在创建自定义 Lambda 函数时，请选择

- 添加允许您的 Lambda 函数执行指定操作和访问的权限Amazon资源。
- 如果您使用提供的 Lambda 函数Amazon IoT Events请选择Python 3.7 运行时选择了 Python 3.7 运行时。

Lambda 函数示例：

```
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 Amazon IAM Identity
 Center (successor to Amazon Single Sign-On). 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')
```

有关更多信息，请参阅 Amazon Lambda 开发人员指南中的什么是 Amazon Lambda？

CloudFormation 模板

使用以下命令 CloudFormation 用于创建 Lambda 函数的模板。

```
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
Amazon IAM Identity Center (successor to Amazon Single Sign-On). 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')
```

使用由提供的 Lambda 函数Amazon IoT Events

在使用由提供的 Lambda 函数时，请选择Amazon IoT Events要管理您的警报通知:

- 您必须在Amaon Sevice (Amazon SEvice (Amazon SESimple Email Service (Amazon SES) 有关更多信息，请参阅 。在 Amazon SES 中验证电子邮件地址，在Amaon Sple Evice 开发者指南.

  如果您收到验证链接，请单击该链接以验证您的电子邮件地址。您也可以检查垃圾邮件文件夹中是否有验证电子邮件。
- 如果您的闹钟发送 SMS 通知，则必须使用 E.164 国际电话号码格式作为电话号码。此格式包含+<country-calling-code><area-code><phone-number>.

电话号码示例：

国家/地区	本地电话号码	E.164 格式化数字
美国	206-555-0100	+12065550100
英国	020-1234-1234	+442012341234
立陶宛	8+601+12345	+37060112345

要查找国家/地区电话代码，请前往国家代码.org.

提供的 Lambda 函数Amazon IoT Events检查您是否使用 E.164 格式的电话号码。但是，它不验证电话号码。如果您确保输入的电话号码准确但没有收到短信通知，则可以联系电话运营商。运营商可能会屏蔽这些消息。

管理收件人

Amazon IoT Events使用Amazon IAM Identity Center (successor to Amazon Single Sign-On)（IAM 身份中心）管理警报收件人对 IAM 身份中心的访问权限。要启用警报向收件人发送通知，您必须启用 IAM 身份中心并将收件人添加到您的 IAM 身份中心存储中。有关更多信息，请参阅 。添加用户在Amazon IAM Identity Center (successor to Amazon Single Sign-On)用户指南.

Important

- 你必须选择相同的Amazon区域Amazon IoT Events,Amazon Lambda和 IAM Identity Center。
- AmazonOrganizations 仅支持一个Amazon Web Services 区域在某一时间。如果您想让 IAM 身份中心在其他区域可用，则必须先删除当前的 IAM 身份中心配置。有关更多信息，请参阅 。Amazon Web Services 区域数据在Amazon IAM Identity Center (successor to Amazon Single Sign-On)用户指南.

Amazon IoT Events 中的安全性

Amazon 十分重视云安全性。作为 Amazon 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Amazon IoT Events 的合规性计划，请参阅合规性计划范围内的 Amazon 服务。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon IoT Events 时应用责任共担模型。以下主题说明如何配置 Amazon IoT Events 以实现您的安全性和合规性目标。你还将学习如何使用其他Amazon服务可以帮助您监控和保护您的Amazon IoT Events资源。

主题

适用于 Amazon IoT Events 的 Identity and Access Management

Amazon Identity and Access Management (IAM) 是一种 Amazon 服务，可以帮助管理员安全地控制对 Amazon 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 Amazon IoT Events 资源。IAM 是一个可以免费使用的 Amazon 服务。

主题

Audience

如何使用 Amazon Identity and Access Management (IAM) 因您可以在 Amazon IoT Events 中执行的操作而异。

服务用户 – 如果您使用 Amazon IoT Events 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Amazon IoT Events 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon IoT Events 中的功能，请参阅 对 Amazon IoT Events 身份和访问进行故障排除 (p. 168)。

服务管理员 – 如果您在公司负责管理 Amazon IoT Events 资源，则您可能具有 Amazon IoT Events 的完全访问权限。您有责任确定您的员工应访问哪些 Amazon IoT Events 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon IoT Events 搭配使用的更多信息，请参阅Amazon IoT Events 如何与 IAM 协同工作 (p. 160)。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon IoT Events 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon IoT Events 基于身份的策略示例，请参阅 Amazon IoT Events 基于身份的策略示例 (p. 163)。

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。有关使用Amazon Web Services Management Console登录的更多信息，请参阅《IAM 用户指南》中的 IAM 控制台和登录页面。

您必须作为Amazon Web Services 账户根用户、IAM 用户或代入 IAM 角色以进行身份验证（登录到 Amazon）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些情况下，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其它公司的凭证访问 Amazon 时，您间接地代入了角色。

要直接登录到Amazon Web Services Management Console，请使用您的密码和根用户电子邮件地址或 IAM 用户名。您可以使用根用户或 IAM 用户访问密钥以编程方式访问 Amazon。Amazon 提供了开发工具包和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。使用 Signature Version 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅《Amazon 一般参考》中的 Signature Version 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其它安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《Amazon 一般参考》中的需要根用户凭证的任务。

IAM 用户和组

IAM 用户是 Amazon 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可能具有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是 Amazon 账户中具有特定权限的实体。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以代入角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 临时 IAM 用户权限 – IAM 用户可以代入 IAM 角色，以暂时获得不同的权限以执行特定的任务。
- 联合用户访问 – 您可以不创建 IAM 用户，而是使用来自 Amazon Directory Service、您的企业用户目录、Web 身份提供商或 IAM Identity Center 身份存储的现有身份。这些身份称为联合身份。要向联合身份分配权限，您可以创建角色并为角色定义权限。当外部身份进行身份验证时，该身份将与角色相关联并被授予其定义的权限。如果您使用 IAM Identity Center，则需要配置权限集。IAM Identity Center 将权限集与 IAM 中的角色相关联，以控制您的身份在进行身份验证后可以访问的内容。有关身份联合验证的更多信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。有关 IAM Identity Center 的更多信息，请参阅《Amazon IAM Identity Center (successor to Amazon Single Sign-On) 用户指南》中的什么是 IAM Identity Center？。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南 中的 IAM 角色与基于资源的策略有何不同。
- Amazon 服务访问 – 服务角色是一个 IAM 角色，服务担任该角色以代表您在您的账户中执行操作。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是否使用 IAM 角色，请参阅 IAM 用户指南 中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 IAM 身份或Amazon资源，以便控制Amazon中的访问。策略是Amazon中的对象；在与身份或资源相关联时，策略定义它们的权限。在某个实体（根用户、IAM 用户或 IAM 角色）发出请求时，Amazon将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概述。

IAM 管理员可以使用策略来指定哪些用户有权访问 Amazon 资源，以及他们可以对这些资源执行哪些操作。每个 IAM 实体（用户或角色）最初没有任何权限。换言之，预设情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、角色或组）的 JSON 权限策略文档。这些策略控制身份可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是可以附加到 Amazon 账户中的多个用户、组和角色的独立策略。托管式策略包括 Amazon 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管式策略与内联策略之间进行选择。

其它策略类型

Amazon 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 Amazon Organizations 中的最大权限。Amazon Organizations 是一项服务，用于分组和集中管理您的企业拥有的多个 Amazon 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体的权限，包括每个 Amazon Web Services 账户 根用户。有关 Organizations 和 SCP 的更多信息，请参阅 Amazon Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 Amazon 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

了解更多信息

有关 Amazon IoT Events 的身份和访问管理的更多信息，请转到以下页面：

- Amazon IoT Events 如何与 IAM 协同工作 (p. 160)
- 对 Amazon IoT Events 身份和访问进行故障排除 (p. 168)

Amazon IoT Events 如何与 IAM 协同工作

在使用 IAM 管理对 Amazon IoT Events 的访问之前，您应了解哪些 IAM 功能可与 Amazon IoT Events 结合使用。要大致了解 Amazon IoT Events 和其他 Amazon 服务如何与 IAM 一起使用，请参阅 IAM 用户指南中的与 IAM 一起使用的 Amazon 服务。

主题

- Amazon IoT Events 基于身份的策略 (p. 160)
- Amazon IoT Events 基于资源的策略 (p. 162)
- 基于 Amazon IoT Events 标签的授权 (p. 162)
- Amazon IoT Events IAM 角色 (p. 162)

Amazon IoT Events 基于身份的策略

使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及指定在什么条件下允许或拒绝操作。Amazon IoT Events 支持特定操作、资源和条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南中的 IAM JSON 策略元素参考。

操作

基于 IAM 身份的策略的 `Action` 元素描述该策略将允许或拒绝的特定操作。策略操作通常与关联的 Amazon API 操作同名。此策略用于策略中以授予执行关联操作的权限。

中的策略操作Amazon IoT Events在操作前使用以下前缀：`iotevents:`. 例如，授予某人创建Amazon IoT Events使用Amazon IoT Events `CreateInput`API 操作，你包括`iotevents:CreateInput`在他们的政策中采取行动。要授予某人发送输入的权限，请使用Amazon IoT Events `BatchPutMessage`API 操作，你包括`iotevents-data:BatchPutMessage`在他们的政策中采取行动。策略语句必须包括 `Action` 或 `NotAction` 元素。Amazon IoT Events 定义了自己的一组操作，这些操作描述了可使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

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

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "iotevents:Describe*"
```

要查看 Amazon IoT Events 操作列表，请参阅 IAM 用户指南中的 Amazon IoT Events 定义的操作。

资源

`Resource` 元素指定要向其应用操作的对象。语句必须包含 `Resource` 或 `NotResource` 元素。您可使用 ARN 来指定资源，或使用通配符 (*) 以指明该语句适用于所有资源。

这些区域有：Amazon IoT Events探测器模型资源拥有以下 ARN：

```
arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/${detectorModelName}
```

有关 ARN 格式的更多信息，请参阅 Amazon Resource Name (ARN) 和 Amazon 服务命名空间。

例如，要指定`Foobar`检测器模型在语句中使用以下 ARN：

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/Foobar"
```

要指定属于特定账户的所有实例，请使用通配符 (*)：

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/*"
```

无法对特定资源执行某些 Amazon IoT Events 操作，例如，用于创建资源的操作。在这些情况下，您必须使用通配符 (*)。

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

一些 Amazon IoT Events API 操作涉及多种资源。例如，`CreateDetectorModel`在其条件语句中引用输入，从而使 IAM 用户必须获得相应权限才能使用输入和检测器模型。要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": [
      "resource1",
```

```
    "resource2"
```

要查看 Amazon IoT Events 资源类型及其 ARN 的列表，请参阅 IAM 用户指南中的 Amazon IoT Events 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon IoT Events 定义的操作。

条件键

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以构建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 Amazon 使用逻辑 `AND` 运算评估它们。如果您为单个条件键指定多个值，则 Amazon 使用逻辑 `OR` 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅 。IAM 策略元素：变量和标签在IAM 用户指南.

Amazon IoT Events 不提供任何特定于服务的条件键，但支持使用某些全局条件键。要查看全部Amazon全局条件键，请参阅Amazon全局条件上下文键在IAM 用户指南。”

示例

要查看 Amazon IoT Events 基于身份的策略的示例，请参阅 Amazon IoT Events 基于身份的策略示例 (p. 163)。

Amazon IoT Events 基于资源的策略

Amazon IoT Events 不支持基于资源的策略。要查看详细的基于资源的策略页面示例，请参阅 https://docs.amazonaws.cn/lambda/latest/dg/access-control-resource-based.html。

基于 Amazon IoT Events 标签的授权

您可以将标签附加到 Amazon IoT Events 资源或将请求中的标签传递到 Amazon IoT Events。要基于标签控制访问，您需要使用 `iotevents:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。有关标记 Amazon IoT Events 资源的更多信息，请参阅给您的 Amazon IoT Events 资源加标签 (p. 186)。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅 查看Amazon IoT Events##基于标签 (p. 165)。

Amazon IoT Events IAM 角色

IAM 角色是 Amazon 账户中具有特定权限的实体。

将临时凭证用于 Amazon IoT Events

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用获取临时安全证书Amazon Security Token Service(Amazon STS) API 操作，例如AssumeRole要么GetFederationToken.

Amazon IoT Events 不支持使用临时凭证。

服务相关角色

服务相关角色允许 Amazon 服务访问其它服务中的资源以代表您完成操作。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon IoT Events 不支持服务相关角色。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其它服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

Amazon IoT Events 支持服务角色。

Amazon IoT Events 基于身份的策略示例

预设情况下，IAM 用户和角色没有创建或修改 Amazon IoT Events 资源的权限。它们还无法使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 执行任务。IAM 管理员必须创建 IAM policy，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南中的在 JSON 选项卡上创建策略。

主题

策略最佳实践

基于身份的策略非常强大。它们确定某个人是否可以创建、访问或删除您账户中的 Amazon IoT Events 资源。这些操作可能会使 Amazon 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 Amazon 托管式策略 – 要快速开始使用 Amazon IoT Events，请使用 Amazon 托管式策略，为您的员工提供他们所需的权限。这些策略已在您的账户中提供，并由 Amazon 维护和更新。有关更多信息，请参阅 IAM 用户指南中的开始使用 Amazon 托管式策略中的权限。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其它权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅《IAM 用户指南》中的授予最低权限。
- 为敏感操作启用 MFA – 为增强安全性，要求 IAM 用户使用多重身份验证 (MFA) 来访问敏感资源或 API 操作。要了解更多信息，请参阅 IAM 用户指南中的在 Amazon 中使用多重身份验证 (MFA)。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 。IAM JSON 策略元素：Condition在IAM 用户指南.

使用 Amazon IoT Events 控制台

要访问 Amazon IoT Events 控制台，您必须拥有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon 账户中的 Amazon IoT Events 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

要确保这些实体仍可使用 Amazon IoT Events 控制台，也可向实体附加以下 Amazon 托管策略。有关更多信息，请参阅 。向用户添加权限在IAM 用户指南：

```
{
    "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}"
        }
    ]
}
```

对于只需要调用 Amazon CLI 或 Amazon API 的用户，无需为其提供最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 Amazon CLI 或 Amazon API 所需的权限。

```
{
       "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": "*"
        }
    ]
}
```

访问一个Amazon IoT Events输入

在本示例中，您希望向您的授予一个 IAM 用户Amazon访问其中一个的帐户Amazon IoT Events输入，`exampleInput`. 您还希望允许用户添加、更新和删除输入。

该策略为用户授予 `iotevents:ListInputs`、`iotevents:DescribeInput`、`iotevents:CreateInput`、`iotevents:DeleteInput` 和 `iotevents:UpdateInput` 权限。有关向用户授予权限并使用控制台测试这些权限的 Amazon S3 服务的示例演练，请参阅演练示例：使用用户策略控制对存储桶的访问.

```
{
   "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/*"
      }
   ]
}
```

查看Amazon IoT Events##基于标签

您可以在基于身份的策略中使用条件，以便基于标签控制对 Amazon IoT Events 资源的访问。此示例演示了如何创建允许查看##. 但是，只有在以下情况下才会授予权限##标签`Owner`的值等于该用户的用户名。此策略还授予在控制台上完成此操作的必要权限。

```
{
    "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}"}
            }
        }
    ]
}
```

您可以将该策略附加到您账户中的 IAM 用户。如果用户名为`richard-roe`尝试查看Amazon IoT Events ## ，##必须标记`Owner=richard-roe`要么`owner=richard-roe`. 否则，他会被拒绝访问。条件标签键 `Owner` 匹配 `Owner` 和 `owner`，因为条件键名称不区分大小写。有关更多信息，请参阅 。IAM JSON 策略元素：Condition在IAM 用户指南.

防止跨服务混淆代理

Note

- 这些区域有：Amazon IoT Events服务仅允许客户使用角色在创建资源的同一账户中启动操作。这意味着无法使用此服务进行混乱的副手攻击。
- 此页面可作为参考，让客户了解混乱的代理问题是如何运作的，如果允许使用跨账户资源，则可以避免Amazon IoT Events服务。

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况，Amazon 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务委托人有权限访问账户中的资源。

我们建议使用资源策略中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键，限制 Amazon IoT Events 为另一项服务提供的资源访问权限。如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文密钥来限制权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 `aws:SourceArn` 值，则 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。

如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。的值`aws:SourceArn`必须是与之关联的探测器型号或警报模型`sts:AssumeRole`请求。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如，`arn:aws:iotevents:*:*:123456789012:*`。

以下示例说明如何将与使用`aws:SourceArn`和`aws:SourceAccount`中的全局条件上下文密钥Amazon IoT Events以防止代理人混淆代理人问题。

主题

示例 1：访问探测器模型

以下角色只能用于访问 DetectorModel 被命名`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"
        }
      }
    }
  ]
 }
}
```

示例 2：访问警报模型

以下角色只能用于访问任何警报模型。

```
{
  "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/*"
        }
      }
    }
  ]
}
```

示例 3：访问指定区域中的资源

以下示例显示了一个角色，您可以使用该角色访问指定区域中的资源。此示例中的区域是*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:*"
        }
      }
    }
  ]
}
```

示例 4: 日志记录选项

要为日志选项提供角色，您需要允许为 IoT Events 中的所有资源代入该角色。因此，必须为资源类型和资源名称使用通配符 (*)。

```
{
  "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:*"
        }
      }
    }
  ]
}
```

对 Amazon IoT Events 身份和访问进行故障排除

使用以下信息可帮助您诊断和修复在使用 Amazon IoT Events 和 IAM 时可能遇到的常见问题。

主题

我无权在 Amazon IoT Events 中执行操作

如果 Amazon Web Services Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

以下示例错误发生在`mateojackson`IAM 用户尝试使用控制台查看有关*##*但没有`iotevents:`*`ListInputs`*权限。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotevents:ListInputs on resource: my-example-input
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `iotevents:`*`ListInput`* 操作访问 *`my-example-input`* 资源。

未授权我执行 iam：PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。请求该人员更新您的策略，以便允许您将角色传递给 Amazon IoT Events。

有些 Amazon 服务允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon IoT Events 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行操作。Mary 不具有将角色传递到服务的权限。

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

在这种情况下，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是管理员并希望允许其他人访问 Amazon IoT Events

要允许其他人访问 Amazon IoT Events，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 Amazon。然后，您必须将策略附加到实体，以便在 Amazon IoT Events 中向其授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我希望允许我的 Amazon 账户之外的人员访问我的 Amazon IoT Events 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon IoT Events 是否支持这些功能，请参阅 Amazon IoT Events 如何与 IAM 协同工作 (p. 160)。
- 要了解如何为您拥有的 Amazon 账户中的资源提供访问权限，请参阅IAM 用户指南中的为您拥有的另一个 Amazon 账户中的 IAM 用户提供访问权限。
- 要了解如何为第三方 Amazon 账户提供您的资源的访问权限，请参阅IAM 用户指南中的 为第三方拥有的 Amazon 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

监控 Amazon IoT Events

监控是保持 Amazon IoT Events 和您的 Amazon 解决方案的可靠性、可用性和性能的重要方面。你应该从你的各个部分收集监控数据Amazon解决方案，以便更轻松地调试出现的多点故障。在开始监控 Amazon IoT Events 之前，您应该创建一个监控计划，其中包括以下问题的答案：

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

下一步，通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 Amazon IoT Events 性能的基准。在监控 Amazon IoT Events 时，存储历史监控数据，以便将此数据与当前性能数据进行比较，确定正常性能模式和性能异常，并设计解决问题的方法。

例如，如果您使用 Amazon EC2，则可以监控实例的 CPU 使用率、磁盘 I/O 和网络使用率。如果性能低于您所建立的基准，则您可能需要重新配置或优化实例，以降低 CPU 利用率、改进磁盘 I/O 或减少网络流量。

主题

- 使用 Amazon IoT Events 记录 Amazon CloudTrail API 调用 (p. 172)

监控工具

Amazon 提供各种可以用来监控 Amazon IoT Events 的工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

自动监控工具

您可以使用以下自动化监控工具来监控 Amazon IoT Events 并在出现错误时报告：

- 亚马逊 CloudWatch 日志— 监控、存储和访问的日志文件Amazon CloudTrail或其他来源。有关更多信息，请参阅 。监控日志文件在亚马逊 CloudWatch 用户指南.
- 亚马逊 CloudWatch 事件— 匹配事件并将事件传送到一个或多个目标函数或流来进行更改、捕获状态信息和采取纠正措施。有关更多信息，请参阅 。什么是 Amazon CloudWatch 事件在亚马逊 CloudWatch 用户指南.
- Amazon CloudTrail日志监控— 在账户之间共享日志文件，监控 CloudTrail 通过将文件发送到，实时记录文件 CloudWatch 记录，用 Java 编写日志处理应用程序，以及验证您的日志文件在交付后未发生更改 CloudTrail. 有关更多信息，请参阅 。使用 CloudTrail 日志文件在Amazon CloudTrail用户指南.

手动监控工具

监控的另一个重要部分Amazon IoT Events涉及手动监控那些物品 CloudWatch 警报不覆盖。这些区域有：Amazon IoT Events, CloudWatch：操作Amazon控制台仪表板提供了 at-a-glance 查看的状态Amazon环境。建议您还要查看 Amazon IoT Events 上的日志文件。

- 该 Amazon IoT Events 控制台显示：
  - 探测器模型数
  - 探测器
  - 输入
  - 设置
- 这些区域有： CloudWatch 主页显示：
  - 当前告警和状态
  - 告警和资源图表
  - 服务运行状况

  此外，您还可以使用 CloudWatch ：
  - 创建自定义控制面板以监控您关心的服务
  - 绘制指标数据图，以排除问题并弄清楚趋势
  - 搜索并浏览您所有的 Amazon 资源指标
  - 创建和编辑警报以接收有关问题的通知

使用 Amazon 监控 CloudWatch

当你开发或调试时Amazon IoT Events detector model, you need to know what Amazon IoT Events正在执行，以及它遇到的任何错误。亚马逊 CloudWatch 监控您的Amazon Web ServicesAmazon) 资源以及您在中运行的应用程序Amazon实时。与 CloudWatch，可全面地的资源使用、应用程序性能和运行状况。启用Amazon CloudWatch 开发时记录Amazon IoT Events探测器模型 (p. 49)包含有关如何启用的信息 CloudWatch 为... 登录Amazon IoT Events. 要生成如下所示的日志，必须设置详细程度“调试” 并提供一个或多个调试目标那是一个探测器型号还有一个可选的KeyValue.

以下示例显示一个 CloudWatch 生成的 DEBUG 级别日志条目Amazon 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
    }
  ]
}
```

使用 Amazon IoT Events 记录 Amazon CloudTrail API 调用

Amazon IoT Events与集成Amazon CloudTrail，是提供用户、角色所采取操作的记录的内容Amazon服务Amazon IoT Events. CloudTrail 捕获Amazon IoT Events作为事件，包括来自Amazon IoT Events控制台和控制台和控制台调用Amazon IoT EventsAPI。

如果您创建了跟踪，则可以启用持续交付 CloudTrail Amazon S3 存储桶的事件，包括的事件Amazon IoT Events. 如果您不配置跟踪，则仍可在 CloudTrail 控制台事件历史. 使用收集的信息 CloudTrail，则可以确定对发出的请求Amazon IoT Events、发出请求的 IP 地址、用户、时间以及其他详细信息。

了解相关更多信息 CloudTrail：参阅阅《Amazon CloudTrail用户指南.

Amazon IoT Events中的 信息 CloudTrail

CloudTrail 已在您的上启用Amazon在您创建账户时，当活动发生在Amazon IoT Events：该活动记录在 CloudTrail 与其他人一起活动Amazon服务事件事件历史. 您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 。查看事件 CloudTrail 事件历史记录.

要持续记录 Amazon 账户中的事件（包括 Amazon IoT Events 的事件），请创建跟踪。跟踪 CloudTrail 将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您还可以配置其他Amazon可进一步分析在中收集的事件数据并采取措施的内容 CloudTrail 日志。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 配置 Amazon SNS 通知 CloudTrail
- 接收 CloudTrail 来自多个区域的日志文件和接收 CloudTrail 来自多个账户的日志文件

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 Amazon 服务发出。

有关更多信息，请参阅 。CloudTrail userIdentity 元素.Amazon IoT Events操作记录在Amazon IoT EventsAPI 参考.

了解 Amazon IoT Events 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。Amazon CloudTrail日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。 CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

当 CloudTrail 您的登录功能已启用Amazon：Amazon IoT Events跟踪操作 CloudTrail 与其他文件一起写入的日志文件Amazon服务记录。 CloudTrail 基于时间段和文件大小来确定何时创建并写入新文件。

每个日志条目都包含有关生成请求的人员的信息。日志条目中的用户身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 Amazon 服务发出。

要在日志文件交付时收到通知，可以配置 CloudTrail 在交付新日志文件时发布 Amazon SNS 通知。有关更多信息，请参阅为 CloudTrail 配置 Amazon SNS 通知。

您也可以将多个 Amazon 区域和多个 Amazon 账户的 Amazon IoT Events 日志文件聚合到单个 Amazon S3 存储桶中。

有关更多信息，请参阅 。接收 CloudTrail 来自多个区域的日志文件和接收 CloudTrail 来自多个账户的日志文件.

以下示例显示一个 CloudTrail 演示以下内容的日志条目`DescribeDetector`操作

```
{
```

```
  "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"
}
```


以下示例显示一个 CloudTrail 演示以下内容的日志条目`CreateDetectorModel`操作


```
{
  "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"
}
```


以下示例显示一个 CloudTrail 演示以下内容的日志条目`CreateInput`操作


```
{
  "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"
}
```


以下示例显示一个 CloudTrail 演示以下内容的日志条目`DeleteDetectorModel`操作


```
{
```

```
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`DeleteInput`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`DescribeDetectorModel`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`DescribeInput`操作

```
{
```

```
  "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"
}
```


以下示例显示一个 CloudTrail 演示以下内容的日志条目`DescribeLoggingOptions`操作


```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`ListDetectorModels`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`ListDetectorModelVersions`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`ListDetectors`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`ListInputs`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`PutLoggingOptions`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`UpdateDetectorModel`操作

```
{
  "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"
}
```

以下示例显示一个 CloudTrail 演示以下内容的日志条目`UpdateInput`操作

```
{
  "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"
}
```

Amazon IoT Events 的合规性验证

作为多个 Amazon 合规性计划的一部分，第三方审核员将评估 Amazon Web Services的安全性与合规性，例如 SOC、PCI、FedRAMP 和 HIPAA。

要了解此服务或其他 Amazon Web Services 是否在特定合规性计划范围内，请参阅合规性计划范围内的 Amazon Web Services。有关常规信息，请参阅Amazon Web Services合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅、在 Amazon Artifact 中下载报告。

您使用 Amazon Web Services 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。Amazon 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署注重安全性和合规性的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计– 该白皮书介绍了公司如何使用 Amazon Web Services 创建符合 HIPAA 标准的应用程序。

  Note

  并非所有 Amazon Web Services都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。
- Amazon合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config 开发人员指南中的使用规则评估资源 – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub：此 Amazon Web Service 提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践规范。

Amazon IoT Events 中的故障恢复能力

Amazon全球基础设施围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施。

Amazon IoT Events 中的基础设施安全性

作为一项托管服务，Amazon IoT Events受Amazon全局网络安全程序，如中所述Amazon Web Services：安全过程概览白皮书。

您可以使用 Amazon 发布的 API 调用通过网络访问 Amazon IoT Events。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service (Amazon STS) 生成临时安全凭证来对请求进行签名。

Amazon IoT Events 配额

这些区域有：Amazon一般参考为以下项提供默认配额Amazon IoT Events对于Amazon账户。除非另有说明，否则，每个配额针对的是Amazon区域。有关更多信息，请参阅 。Amazon IoT Events终端节点和配额和AmazonService Quotas在里面Amazon一般参考.

要请求提高服务配额，请在Support 中心控制台。有关更多信息，请参阅 Service Quotas 用户指南中的请求增加服务限额。

Note

- 账户中的探测器模型和输入的所有名称必须是唯一的。
- 在创建探测器模型和输入后，您无法更改其名称。

给您的 Amazon IoT Events 资源加标签

为了帮助您管理和组织探测器模型和输入，您可以选择将自己的元数据以标签的形式分配给其中每个资源。本部分介绍标签并说明如何创建标签。

有关标签的基本知识

标签可让您按各种标准（例如用途、所有者或环境）对 Amazon IoT Events 资源进行分类。这在您有许多相同类型的资源时会非常有用。您可以根据分配到特定资源的标签来快速识别该资源。

每个标签都包含您定义的一个键和一个可选值。例如，您可以为输入定义一组标签，以跟踪发送这些输入的设备。我们建议您为每类资源创建一组可满足您的需求的标签键。使用一组连续的标签键，管理资源时会更加轻松。

您可以根据添加或应用的标签搜索和筛选资源，使用标签对成本进行分类和跟踪，还可以使用标签控制对资源的访问权限，如中所述在 IAM 策略中使用标签在Amazon IoT开发人员指南.

为便于使用，使用中的标签编辑器Amazon Web Services Management Console提供了一种用于创建和管理标签的集中而统一的方法。有关更多信息，请参阅 。使用标签编辑器在在上使用Amazon Web Services Management Console.

您也可以使用 Amazon CLI 和 Amazon IoT Events API 处理标签。在创建标签时，您可以使用以下方法将标签与探测器模型和输入相关联`"Tags"`字段（采用格式）

- CreateDetectorModel
- CreateInput

您可以使用以下命令为支持标记的现有资源添加、修改或删除标签：

- TagResource
- ListTagsForResource
- UntagResource

您可以修改标签的密钥和值，还可以随时删除资源的标签。您可以将标签的值设为空的字符串，但是不能将其设为空值。如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。如果删除资源，则所有与资源相关的标签都将被删除。

界面中会提供更多信息Amazon标签策略.

标签限制

下面是适用于标签的基本限制：

- 每个资源的标签数上限 – 50
- 最大键长度 — 127 个 Unicode 字符（采用 UTF-8 格式）
- 最大值长度 — 255 个 Unicode 字符（采用 UTF-8 格式）

- 标签键和值区分大小写。
- 请勿使用"aws:"在标签名称或值中使用前缀，因为它专为Amazon使用。您无法编辑或删除带此前缀的标签名称或值。具有此前缀的标签不计入每个资源的标签数限制。
- 如果您的标记模式针对多个服务和资源使用，请记得其它服务可能对允许使用的字符有限制。通常允许使用的字符包括：可用 UTF-8 格式表示的字母、空格和数字以及以下特殊字符：+ - = . _ : / @。

在 IAM 策略中使用标签

可以在用于 Amazon IoT Events API 操作的 IAM 策略中应用基于标签的资源级权限。这可让您更好地控制用户可创建、修改或使用哪些资源。

在 IAM 策略中将 `Condition` 元素（也称作 `Condition` 块）与以下条件上下文键和值结合使用来基于资源标签控制用户访问（权限）：

- 使用 `aws:ResourceTag/<tag-key>: <tag-value>` 可允许或拒绝带特定标签的资源上的用户操作。
- 使用 `aws:RequestTag/<tag-key>: <tag-value>` 可要求在发出创建或修改允许标签的资源的 API 请求时使用（或不使用）特定标签。
- 使用 `aws:TagKeys: [<tag-key>, ...]` 可要求在发出创建或修改允许标签的资源的 API 请求时使用（或不使用）一组特定标签键。

> **Note**
>
> IAM 策略中的条件上下文键和值仅适用于能够标记的资源的标识符是必需参数的那些 Amazon IoT Events 操作。

使用标签控制访问在Amazon Identity and Access Management用户指南包含有关使用标签的更多信息。这些区域有：IAM JSON 策略参考该指南的一部分包含 IAM 中的 JSON 策略的元素、变量和评估逻辑的详细语法、描述和示例。

以下示例策略应用两个基于标签的限制。受此策略限制的 IAM 用户：

- 无法为资源提供标签“env=prod”（在示例中，请参阅行 `"aws:RequestTag/env" : "prod"`
- 无法修改或访问具有现有标签“env=prod”的资源（在示例中，请参阅行 `"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": "*"
        }
    ]
}
```

您还可以通过将给定标签键包含在列表中来为它们指定多个标签值，如下所示。

```
            "StringEquals" : {
              "aws:ResourceTag/env" : ["dev", "test"]
            }
```

Note

如果您基于标签允许或拒绝用户访问资源，则必须考虑显式拒绝用户对相同资源添加或删除这些标签的能力。否则，用户可能通过修改资源标签来绕过您的限制并获得资源访问权限。

排除 Amazon IoT Events 的故障

使用以下部分中的信息来排查和解决 Amazon IoT Events 的问题。

主题

常见Amazon IoT Events问题和解决方案

请参阅以下部分以解决错误并找到可能的解决方案来解决以下问题Amazon IoT Events.

错误

探测器模型创建错误

尝试创建探测器模型时出现错误。

解决方案

当您创建探测器模型时，必须考虑以下限制。

- 每个操作只允许一个操作`action`字段中返回的子位置类型。
- 这些区域有：`condition`对于以下应用程序的必需`transitionEvents`. 它是可选的`OnEnter`,`OnInput`，以及`OnExit`事件。
- 如果`condition`字段为空，条件表达式的计算结果等于`true`.
- 条件表达式的计算结果应为布尔值。如果结果不是布尔值，则等效于`false`而且不会触发`actions`或者过渡到`nextState`在事件中指定。

有关更多信息，请参阅 探测器模型限制 (p. 61)。

来自已删除探测器模型的更新

我在几分钟前更新或删除了探测器模型，但我仍然通过 MQTT 消息或 SNS 警报从旧探测器模型获得状态更新。

解决方案

如果您更新、删除或重新创建探测器模型（请参见UpdateDetectorModel），在删除所有探测器实例并使用新模型之前会有一段延迟。在此期间，输入可能会继续由先前版本的探测器模型的实例处理。您可能会继续收到由先前探测器模型定义的警报。等待至少七分钟，然后再重新检查更新或报告错误。

动作触发失败（满足条件时）

当条件满足时，检测器无法触发动作或过渡到新状态。

解决方案

验证检测器条件表达式的计算结果是否为布尔值。如果结果不是布尔值，则等效于`false`而且不会触发`action`或者过渡到`nextState`在事件中指定。有关更多信息，请参阅 。条件表达式语法.

动作触发失败（突破阈值时）

当条件表达式中的变量达到指定值时，检测器不会触发动作或事件转换。

解决方案

如果你更新`setVariable`为了`onInput`,`onEnter`，或`onExit`，计算任何值时都不使用新值`condition`在当前的处理周期内。取而代之的是，在当前周期结束之前使用原始值。您可以通过将设置以下应用程序来更改此行为`evaluationMethod`探测器模型定义中的参数。当`evaluationMethod`已设置为`SERIAL`，变量按定义事件的顺序更新并评估事件条件。当`evaluationMethod`已设置为`BATCH`（默认），只有在评估了所有事件条件之后，才会更新变量并执行事件。

状态使用不正确

当我尝试使用向输入发送消息时，检测器进入了错误的状态`BatchPutMessage`.

解决方案

如果您使用BatchPutMessage要向输入发送多条消息，则无法保证消息或输入的处理顺序。为保证订购，一次发送一条消息，每次都等待`BatchPutMessage`以表彰成功。

连接消息

我得到一个`('Connection aborted.', error(54, 'Connection reset by peer'))`当我尝试调用或调用 API 时出错。

解决方案

验证 OpenSSL 是否使用 TLS 1.1 或更高版本来建立连接。这应该是大多数 Linux 发行版或 Windows 版本 7 及更高版本下的默认设置。macOS 的用户可能需要升级 OpenSSL。

InvalidRequestException 消息

我明白了 InvalidRequestException 当我尝试打电话时`CreateDetectorModel`和`UpdateDetectorModel`API。

解决方案

检查以下几点以帮助解决问题。有关更多信息，请参阅 。CreateDetectorModel和UpdateDetectorModel.

- 确保不要同时使用两者seconds和durationExpression作为参数SetTimerAction同时。
- 请确保您的字符串表达式为durationExpression有效。字符串表达式可以包含数字、变量 ($variable.<variable-name>) 或输入值 ($input.<input-name>.<path-to-datum>)。

亚马逊 CloudWatch 日志action.setTimer错误

你可以设置亚马逊 CloudWatch 要监控的日志Amazon IoT Events探测器模型实例。以下是生成的常见错误 Amazon IoT Events，当您使用以下应用程序时action.setTimer.

- 错误 您的计时器时长表达式名为*<timer-name>*无法计算为数字。

解决方案

请确保您的字符串表达式为durationExpression可以转换为数字。不允许使用其他数据类型，例如布尔值。

- 错误 名为的计时器的持续时间表达式的计算结果*<timer-name>*大于 31622440。为确保准确性，请确保您的持续时间表达式指的是介于 60-31622400 之间的值。

解决方案

请确保计时器的持续时间小于或等于31622400秒。持续时间的计算结果向下舍入为最接近的整数。

- 错误 名为的计时器的持续时间表达式的计算结果*<timer-name>*小于 60。为确保准确性，请确保您的持续时间表达式指的是介于 60-31622400 之间的值。

解决方案

请确保计时器的持续时间大于或等于60秒。持续时间的计算结果向下舍入为最接近的整数。

- 错误 您的计时器时长表达式名为*<timer-name>*无法评估。检查变量名称、输入名称和数据路径，确保引用的是现有变量和输入。

解决方案

确保您的字符串表达式引用现有变量和输入。字符串表达式可以包含数字、变量 ($variable.*variable-name*) 和输入值 ($input.*input-name*.*path-to-datum*)。

- 错误 无法设置名为的计时器<timer-name>. 检查您的持续时间表达式，然后重试。

解决方案

请参阅SetTimerAction操作以确保您指定了正确的参数，然后再次设置计时器。

有关更多信息，请参阅 。启用亚马逊 CloudWatch 开发时记录Amazon IoT Events探测器模型.

亚马逊 CloudWatch 负载

你可以设置亚马逊 CloudWatch 要监控的日志Amazon IoT Events探测器模型实例。以下是生成的常见的错误和警告Amazon IoT Events，当您配置操作负载。

- 错误 我们无法评估你对这个动作的表情。确保变量名称、输入名称和数据路径引用现有变量和输入值。此外，请验证负载的大小是否小于 1 KB，即有效负载允许的最大容量。

解决方案

确保输入正确的变量名称、输入名称和数据路径。如果操作负载大于 1 KB，您也可能会收到此错误消息。

- 错误 我们无法解析你的内容表达式以获取有效负载*<action-type>*. 使用正确的语法输入内容表达式。

解决方案

内容表达式可以包含字符串 (`'string'`)、变量 (`$variable.variable-name`)，输入值 (`$input.input-name.path-to-datum`)、字符串连接和包含的字符串`${}`.

- 错误 你的负载表达式 {`##`} 无效。定义的负载类型为 JSON，因此您必须指定一个符合以下条件的表达式Amazon IoT Events会计算为一个字符串。

解决方案

如果指定的负载类型为 JSON，Amazon IoT Events首先检查服务是否可以将您的表达式计算为字符串。评估结果不能是布尔值或数字。如果验证失败，您可能会收到此错误。

- 警告：操作已执行，但我们无法将操作负载的内容表达式评估为有效 JSON。定义的负载类型是 JSON。

解决方案

确保此Amazon IoT Events如果您将负载类型定义为，则可以将操作负载的内容表达式评估为有效的JSONJSON.Amazon IoT Events运行动作，即使Amazon IoT Events无法将内容表达式评估为有效的JSON。

有关更多信息，请参阅 。启用亚马逊 CloudWatch 开发时记录Amazon IoT Events探测器模型.

数据类型不兼容

Message 数据类型不兼容 [<inferred-types>] 已找到<reference>在以下表达式中：<expression>

解决方案

您可能会收到此错误，原因以下之一：

- 您的引用的评估结果与表达式中的其他操作数不兼容。
- 不支持传递给函数的参数的类型。

当您在表达式中使用引用时，请检查以下几点：

- 当您使用引用作为具有一个或多个运算符的操作数时，请确保您引用的所有数据类型都兼容。

  例如，在以下表达式中，INTEGER2是两个的操作数==和&&运算符。为了确保操作数兼容，`$variable.testVariable + 1`和`$variable.testVariable`必须引用整数或小数。

  此外，INTEGER1是的操作数+运算符。因此`$variable.testVariable`必须引用整数或小数。

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- 当您使用引用作为传递给函数的参数时，请确保该函数支持您引用的数据类型。

  例如以下内容`timeout("time-name")`函数需要一个以双引号作为参数的字符串。如果你使用参考来表示`#####`值，则必须引用带双引号的字符串。

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

> **Note**
>
> 对于`convert(type, expression)`函数，如果你使用引用来表示`##`值，您的参考文献的评估结果必须为`String`,`Decimal`，或`Boolean`.

有关更多信息，请参阅参考 (p. 101)。

通过运行分析对探测器模型进行故障排除

Amazon IoT Events可以分析您的探测器模型并生成分析结果，而无需向探测器模型发送输入数据。Amazon IoT Events执行本节所述的一系列分析以检查您的探测器模型。此高级故障排除解决方案还汇总了诊断信息，包括严重级别和位置，以便您可以快速找到并修复探测器模型中的潜在问题。有关探测器型号的诊断错误类型和消息的更多信息，请参见探测器模型分析和诊断信息 (p. 193).

您可以使用Amazon IoT Events控制台，API,Amazon Command Line Interface(Amazon CLI)，或Amazon开发工具包查看探测器模型分析产生的诊断错误消息。

Note

- 在发布探测器模型之前，必须修复所有错误。
- 我们建议您在生产环境中使用探测器模型之前查看警告并采取必要措施。否则，探测器模型可能无法按预期运行。
- 您最多可以在 10 个分析中进行分析`RUNNING`同时处于状态。

若要了解如何分析探测器模型，请参阅分析探测器模型（控制台） (p. 199)要么分析探测器模型 (Amazon CLI) (p. 200).

主题

- 探测器模型分析和诊断信息 (p. 193)
- 分析探测器模型（控制台） (p. 199)
- 分析探测器模型 (Amazon CLI) (p. 200)

探测器模型分析和诊断信息

探测器模型分析收集以下诊断信息：

- Level— 分析结果的严重性级别。根据严重性级别，分析结果分为三大类：
  - 信息(`INFO`) — 信息结果告诉您探测器模型中的重要场。这种类型的结果通常不需要立即采取行动。
  - 警告(`WARNING`) — 警告结果特别提请注意可能导致探测器模型出现问题的字段。我们建议您在生产环境中使用探测器模型之前查看警告并采取必要措施。否则，探测器模型可能无法按预期运行。
  - 错误(`ERROR`) — 错误结果会通知您在探测器模型中发现了问题。Amazon IoT Events当你尝试发布探测器模型时，会自动执行这组分析。必须先修复所有错误，然后才能发布探测器模型。
- 位置— 包含可用于在探测器模型中定位分析结果所引用的场的信息。位置通常包括状态名称、过渡事件名称、事件名称和表达式（例如，`in state TemperatureCheck in onEnter in event Init in action setVariable`)。
- 类型— 分析结果的类型。分析类型可分为以下几类：
  - `supported-actions`–Amazon IoT Events可以在检测到指定事件或过渡事件时调用操作。您可以定义内置动作以使用计时器或设置变量，或将数据发送到其他计时器Amazon服务。您必须指定与其他操作配合使用的操作Amazon中的服务Amazon所在的地区Amazon服务可用。
  - `service-limits`— 服务配额（也称为限制）是您使用的服务资源或操作的最大或最小数量Amazon账户。除非另有说明，否则，每个配额是区域特定的。根据您的业务需求，您可以更新探测器模型以避免遇到限制或申请增加配额。您可以请求增加某些配额，但其他一些配额无法增加。有关更多信息，请参阅 配额。
- **`structure`**— 探测器模型必须包含所有必需的组件，例如状态，并遵循以下结构Amazon IoT Events支持。探测器模型必须至少具有一种状态和一个条件，用于评估传入的输入数据以检测重大事件。当检测到

事件时，探测器模型转换到下一个状态并可以调用操作。这些事件被称为过渡事件。过渡事件必须引导下一个状态进入。

- **expression-syntax**–Amazon IoT Events在创建和更新探测器模型时提供了多种指定值的方法。您可以在表达式中使用文字、运算符。您可以使用表达式来指定文字值，或者Amazon IoT Events可以在指定特定值之前对表达式求值。您的表达式必须遵循所需的语法。有关更多信息，请参阅 表达式 (p. 97)。

  中的探测器模型表达式Amazon IoT Events可以引用特定的数据或资源。

  - **data-type**–Amazon IoT Events支持整数、十进制、字符串和布尔数据类型。如果Amazon IoT Events可以在表达式求值期间自动将一种数据类型的数据转换为另一种数据类型，这些数据类型是兼容的。

    Note

    - 整数和十进制是支持的唯一兼容数据类型Amazon IoT Events.
    - Amazon IoT Events无法计算算术表达式，因为Amazon IoT Events无法将整数转换为字符串。
  - **referenced-data**— 必须先定义探测器模型中引用的数据，然后才能使用这些数据。例如，如果要向 DynamoDB 表发送数据，则必须定义一个引用该表名的变量，然后才能在表达式中使用该变量 (`$variable.TableName`)。
  - **referenced-resource**— 探测器模型使用的资源必须可用。必须先定义资源，然后才能使用它们。例如，您要创建探测器模型以监控温室的温度。你必须定义一个输入 (`$input.TemperatureInput`) 将传入的温度数据传送到您的探测器模型，然后才能使用`$input.TemperatureInput.sensorData.temperature`以参考温度。

请参阅以下部分，通过分析探测器模型来排除错误并找到可能的解决方案。

探测器模型错误疑难解答

上述错误类型提供有关探测器模型的诊断信息，并对应于您可能检索到的消息。使用这些消息和建议的解决方案来排除探测器模型的错误。

消息和解决方案

- Location (p. 194)
- supported-actions (p. 194)
- service-limits (p. 195)
- structure (p. 195)
- expression-syntax (p. 197)
- data-type (p. 198)
- referenced-data (p. 199)
- referenced-resource (p. 199)

Location

包含以下信息的分析结果`Location`，对应于以下错误消息：

- Message— 包含有关分析结果的其他信息。这可以是信息、警告或错误消息。

  解决方案：如果您指定的操作符合以下条件，则可能会收到此错误消息Amazon IoT Events目前不支持。有关受支持的操作的列表，请参阅支持的操作 (p. 87).

supported-actions

包含以下信息的分析结果`supported#actions`，对应于以下错误消息：

- 消息: 动作定义中存在无效的操作类型：*####*.

  解决方案：如果您指定的操作符合以下条件，则可能会收到此错误消息Amazon IoT Events目前不支持。有关受支持的操作的列表，请参阅支持的操作 (p. 87).
- 消息: DetectorModel 定义有一个*aws-service*行动，但是*aws-service*该地区不支持服务*####*.

  解决方案：如果您指定的操作受支持，则可能会收到此错误消息Amazon IoT Events，但此操作在您的当前区域中不可用。当你尝试将数据发送到时，可能会发生这种情况Amazon该区域中不可用的服务。您还必须为两者选择相同的区域Amazon IoT Events还有Amazon您正在使用的服务。

service-limits

包含以下信息的分析结果`service#limits`，对应于以下错误消息：

- 消息: 有效负载中允许的内容表达式超过了限制*content-expression-size*事件中的字节数*Event ##*状态*####*.

  解决方案：如果您的操作负载的内容表达式大于 1024 字节，您可能会收到此错误消息。负载的内容表达式的大小最高可达 1024 字节。
- 消息: 探测器模型定义中允许的状态数量超过了限制*states-per-detector-model*.

  解决方案：如果您的探测器模型的状态超过 20 个，则可能会收到此错误消息。探测器模型最多可拥有 20 个状态。
- 消息: 计时器的持续时间*#####*应为至少*minimum-timer-duration*长达数秒。

  解决方案：如果计时器的持续时间少于 60 秒，则可能会收到此错误消息。我们建议计时器的持续时间介于 60 到 31622400 秒之间。如果您为计时器持续时间指定了一个表达式，则持续时间表达式的计算结果将向下舍入到最近的整数。
- 消息: 每个事件允许的操作数量超过了限制*actions-per-event*在探测器模型定义中

  解决方案：如果事件的操作超过 10 个，您可能会收到此错误消息。在探测器模型中，每个事件最多可拥有 10 个操作。
- 消息: 每个州允许的过渡事件数量超过了限制*transition-events-per-state*在探测器模型定义中。

  解决方案：如果状态的过渡事件超过 20 个，您可能会收到此错误消息。探测器模型中的每个状态最多可拥有 20 个过渡事件。
- 消息: 每个州允许的事件数超过了限制*events-per-state*在探测器模型定义中

  解决方案：如果该州有超过 20 个事件，您可能会收到此错误消息。探测器模型中的每个状态最多可拥有 20 个事件。
- 消息: 可以与单个输入关联的最大探测器模型数量可能已达到限制。输入*####*在中使用*detector-models-per-input*探测器模型路线。

  解决方案：如果您尝试将输入路由到 10 个以上的探测器模型，则可能会收到此警告消息。您最多可以将 10 个不同的探测器模型与单个探测器模型相关联。

structure

包含以下信息的分析结果`structure`，对应于以下错误消息：

- 消息: 操作可能只定义了一种类型，但找到的操作为*number-of-types*类型。请分成单独的操作。

  解决方案：如果您使用 API 操作创建或更新探测器模型，在单个字段中指定了两个或多个操作，则可能会收到此错误消息。您可以定义一个数组`Action`对象。确保将每个动作定义为一个单独的对象。
- 消息: 这些区域有： TransitionEvent *transition-event-name*过渡到不存在的状态*####*.

解决方案：如果出现以下情况，您可能会收到此错误消息Amazon IoT Events找不到你的过渡事件引用的下一个状态。确保定义了下一个状态并且输入了正确的州名。

- 消息: 这些区域有： DetectorModelDefinition 有一个共享的状态名称：找到状态*####*和*number-of-states*重复。

  解决方案：如果您为一个或多个州使用相同的名称，则可能会收到此错误消息。确保为探测器模型中的每个状态指定一个唯一名称。州名必须包含 1-128 个字符。有效字符：a-z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-

- 消息: 定义的 initialStateName *initial-state-name*不对应于一个定义的国家。

  解决方案：如果初始状态名称不正确，您可能会收到此错误消息。在输入到达之前，探测器模型保持初始（开始）状态。输入到达后，探测器模型立即转换到下一个状态。确保初始状态名称是已定义状态的名称，并且输入了正确的名称。

- 消息: 探测器模型定义必须在一个条件中使用至少一个输入。

  解决方案：如果您没有在条件中指定输入，则可能会收到此错误。必须在至少一个条件中使用至少一个输入。否则，Amazon IoT Events不评估传入的数据。

- 消息: 只能在秒中设置一秒钟和 durationExpression SetTimer.

  解决方案：如果您同时使用两者，则可能会收到此错误消息`seconds`和`durationExpression`为您的计时器。请确保使用以下任一项`seconds`要么`durationExpression`作为的参数`SetTimerAction`. 有关更多信息，请参阅《Amazon IoT Events API 参考》中的 SetTimerAction。

- 消息: 您的探测器模型中的动作无法实现。检查启动操作的条件。

  解决方案：如果探测器模型中的某个动作无法触及，则该事件的条件评估为 false。检查包含操作的事件的状况，确保其计算结果为 true。当事件的条件计算为 true 时，该操作应可访问。

- 消息: 正在读取输入属性，但这可能是由计时器过期造成的。

  解决方案：出现以下任一情况时，可以读取输入属性的值：

  - 已收到新的输入值。
  - 当探测器中的计时器过期时。

  为确保仅在接收到输入的新值时才计算该输入属性，请调用`triggerType(“Message”)`在你的条件下起作用如下：

  探测器模型中正在评估的原始条件：

  ```
  if ($input.HeartBeat.status == “OFFLINE”)
  ```

  将变得类似于以下内容：

  ```
  if ( triggerType("MESSAGE") &&  $input.HeartBeat.status == “OFFLINE”)
  ```

  在哪里打电话给`triggerType(“Message”)`函数出现在条件中提供的初始输入之前。通过使用这种技术，`triggerType("Message")`函数的计算结果为 true 并满足接收新输入值的条件。有关使用方法的更多信息`triggerType`函数，搜索`triggerType`在里面表达式部分中的部分Amazon IoT Events开发人员指南

- 消息: 探测器模型中的状态无法到达。检查将导致过渡到所需状态的条件。

  解决方案：如果探测器模型中的某个状态无法到达，则导致传入过渡到该状态的条件的计算结果为 false。检查探测器模型中传入转换为不可达状态的条件是否为真，以便可以达到所需的状态。

- 消息: 计时器过期可能会导致发送意外数量的消息。

解决方案：为了防止您的探测器模型进入无限状态，即由于计时器过期而发送意外数量的消息，请考虑调用`triggerType("Message")`函数，在探测器模型的条件下，如下所示：

探测器模型中正在评估的原始条件：

```
if (timeout("awake"))
```

将转化为类似于以下内容：

```
if (triggerType("MESSAGE") && timeout("awake"))
```

在哪里打电话给`triggerType("Message")`函数出现在条件中提供的初始输入之前。

此更改可防止在探测器中启动计时器操作，从而防止无限循环发送消息。有关如何在探测器中使用计时器操作的更多信息，请参阅使用内置操作的页面Amazon IoT Events开发人员指南

expression-syntax

包含以下信息的分析结果`expression#syntax`，对应于以下错误消息：

- 消息: 你的负载表达式 {*##*} 无效。定义的负载类型为 JSON，因此您必须指定一个表达式Amazon IoT Events会计算为一个字符串。

  解决方案：如果指定的负载类型为 JSON，Amazon IoT Events首先检查服务是否可以将您的表达式计算为字符串。评估结果不能是布尔值或数字。如果验证不成功，您可能会收到此错误。
- 消息: `SetVariableAction.value`必须是表达式。解析值 '失败*###*'

  解决方案：您可以使用`SetVariableAction`用 a 定义一个变量`name`和`value`. 这些区域有：`value`可以是字符串、数字或布尔值。您也可以为以下内容指定一个表达式`value`. 有关更多信息，请参阅 。SetVariableAction，在Amazon IoT EventsAPI 参考.
- 消息: 我们无法解析你对属性的表达式 (*####*) 用于 DynamoDB 操作。使用正确的语法输入表达式。

  解决方案：必须对中的所有参数使用表达式`DynamoDBAction`。替换模板。有关更多信息，请参阅 。DynamoDBAction在里面Amazon IoT EventsAPI 参考.
- 消息: 我们无法解析你对 DynamoDBv2 操作的 tableName 的表达式。使用正确的语法输入表达式。

  解决方案：这些区域有：`tableName`在`DynamoDBv2Action`必须是字符串。必须使用表达式表示`tableName`. 这些表达式接受文字、运算符、函数、引用和替代模板。有关更多信息，请参阅 。DynamoDBv2Action在里面Amazon IoT EventsAPI 参考.
- 消息: 我们无法将您的表达式评估为有效的 JSON。DynamoDBv2 操作仅支持 JSON 负载类型。

  解决方案：的负载类型`DynamoDBv2`必须是 JSON。确保已存在Amazon IoT Events可以评估您的内容表达式，将有效负载转换为有效的 JSON。有关更多信息，请参阅 。DynamoDBv2Action，在Amazon IoT EventsAPI 参考.
- 消息: 我们无法解析你的内容表达式以获取有效负载*####*. 使用正确的语法输入内容表达式。

  解决方案：内容表达式可以包含字符串 ('*#*')、变量 ($variable。*####*)、输入值 ($input。*####.path-to-datum*)、字符串连接和包含的字符串`${}`.
- 消息: 自定义的负载必须为非空。

  解决方案：如果您选择，则可能会收到此错误消息自定义有效负载用于你的操作，但没有在里面输入内容表达式Amazon IoT Events控制台。如果您选择自定义有效负载，您必须在下方输入内容表达式自定义有效负载. 有关更多信息，请参阅 。Payload在里面Amazon IoT EventsAPI 参考.
- 消息: 无法解析持续时间表达式 '*Duration ###*'用于计时器'*#####*'。

解决方案：计时器的持续时间表达式的计算结果必须是 60—31622400 之间的值。持续时间的计算结果向下舍入为最接近的整数。

- 消息: 解析表达式 '失败*##*'对于*####*

  解决方案：如果指定操作的表达式语法不正确，您可能会收到此消息。确保输入的表达式语法正确。有关更多信息，请参阅 语法 (p. 97)。

- 消息: 你的*fieldName*为了IotSitewiseAction无法解析。在表达式中必须使用正确的语法。

  解决方案：如果出现以下情况，您可能会收到此错误Amazon IoT Events无法解析你的*fieldName*为了IotSitewiseAction. 确保*fieldName*使用以下表达式Amazon IoT Events可以解析。有关更多信息，请参阅《Amazon IoT Events API 参考》中的 IotSiteWiseAction。

data-type

包含以下信息的分析结果data#type，对应于以下错误消息：

- 消息: 持续时间表达式*Duration ###*用于计时器*#####*无效，它必须返回一个数字。

  解决方案：如果出现以下情况，您可能会收到此错误消息Amazon IoT Events无法将计时器的持续时间表达式计算为数字。确保您的durationExpression可以转换为数字。不支持其他数据类型，如布尔值。

- 消息: 表达式*#####*不是有效的条件表达式。

  解决方案：如果出现以下情况，您可能会收到此错误消息Amazon IoT Events无法评估你的condition-expression为一个布尔值。布尔值必须是以下任一值TRUE要么FALSE. 确保您的条件表达式可以转换为布尔值。如果结果不是布尔值，则等效于FALSE而且不会调用动作或过渡到nextState在事件中指定。

- 消息: 数据类型不兼容 [*####*] 已找到为*##*在以下表达式中：*##*

  解决方案：解决方案：探测器模型中相同输入属性或变量的所有表达式都必须引用相同的数据类型。

  使用以下信息解决此问题：

  - 当您使用引用作为具有一个或多个运算符的操作数时，请确保您引用的所有数据类型都兼容。

    例如，在以下表达式中，整数2是两个的操作数==和&&运算符。为了确保操作数兼容，$variable.testVariable + 1和$variable.testVariable必须引用整数或小数。

    此外，INTEGER1是的操作数+运算符。因此，$variable.testVariable必须引用整数或小数。

    ```
    '$variable.testVariable + 1 == 2 && $variable.testVariable'
    ```

  - 当您使用引用作为传递给函数的参数时，请确保该函数支持您引用的数据类型。

    例如，以下timeout("*time-name*")函数需要一个以双引号作为参数的字符串。如果你使用参考来表示*#####*值，则必须引用带双引号的字符串。

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

    > **Note**
    >
    > 对于convert(*type*, *expression*)函数，如果你使用引用来表示*##*值，您的参考文献的评估结果必须为String,Decimal，或Boolean.

  有关更多信息，请参阅 参考 (p. 101)。

- 消息: 数据类型不兼容 [*####*] 将与结合使用*##*. 这可能会导致运行时错误。

解决方案：如果同一个输入属性或变量的两个表达式引用了两种数据类型，您可能会收到此警告消息。确保相同输入属性或变量的表达式在探测器模型中引用相同的数据类型。

referenced-data

包含以下信息的分析结果`referenced#data`，对应于以下错误消息：

- 消息: 检测到计时器损坏：计时器*#####*在表达式中使用，但从未设置过。

  解决方案：如果您使用未设置的计时器，则可能会收到此错误消息。在表达式中使用计时器之前，必须先设置计时器。另外，请确保输入正确的计时器名称。
- 消息: 检测到变量已损坏：变量*####*在表达式中使用，但从未设置过。

  解决方案：如果您使用未设置的变量，则可能会收到此错误消息。在表达式中使用变量之前，必须先对其进行设置。另外，请确保输入正确的变量名称。
- 消息: 检测到损坏的变量：在将变量设置为值之前，先在表达式中使用变量。

  解决方案：必须先为每个变量赋值，然后才能在表达式中对其进行计算。在每次使用之前设置变量的值，以便可以检索其值。另外，请确保输入正确的变量名称。

referenced-resource

包含以下信息的分析结果`referenced#resource`，对应于以下错误消息：

- 消息: 探测器模型定义包含对不存在的输入的引用。

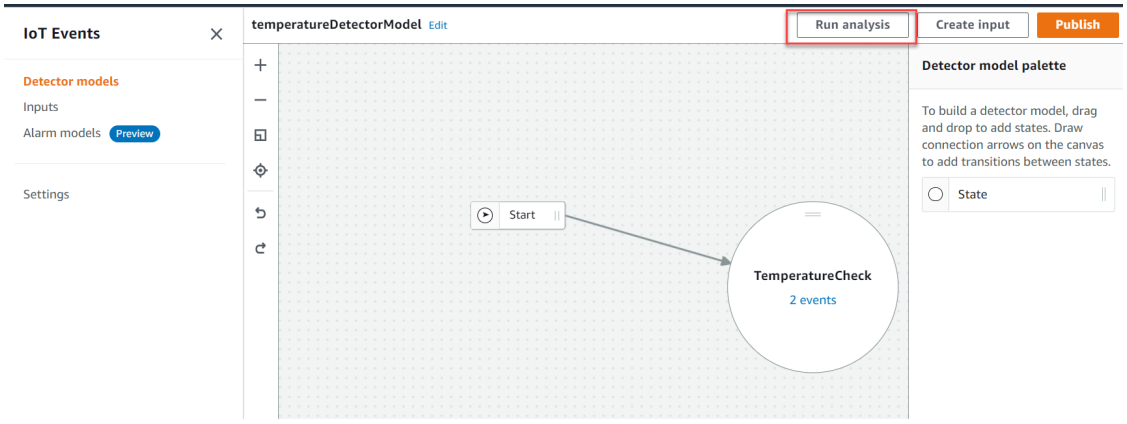
  解决方案：如果您使用表达式引用不存在的输入，则可能会收到此错误消息。确保您的表达式引用现有输入并输入正确的输入名称。如果您没有输入，请先创建一个输入。
- 消息: 探测器模型定义包含无效 InputName：*####*

  解决方案：如果您的探测器型号包含无效的输入名称，您可能会收到此错误消息。确保输入正确的输入名称。输入名称必须包含 1-128 个字符。有效字符：a-z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-Z、A-

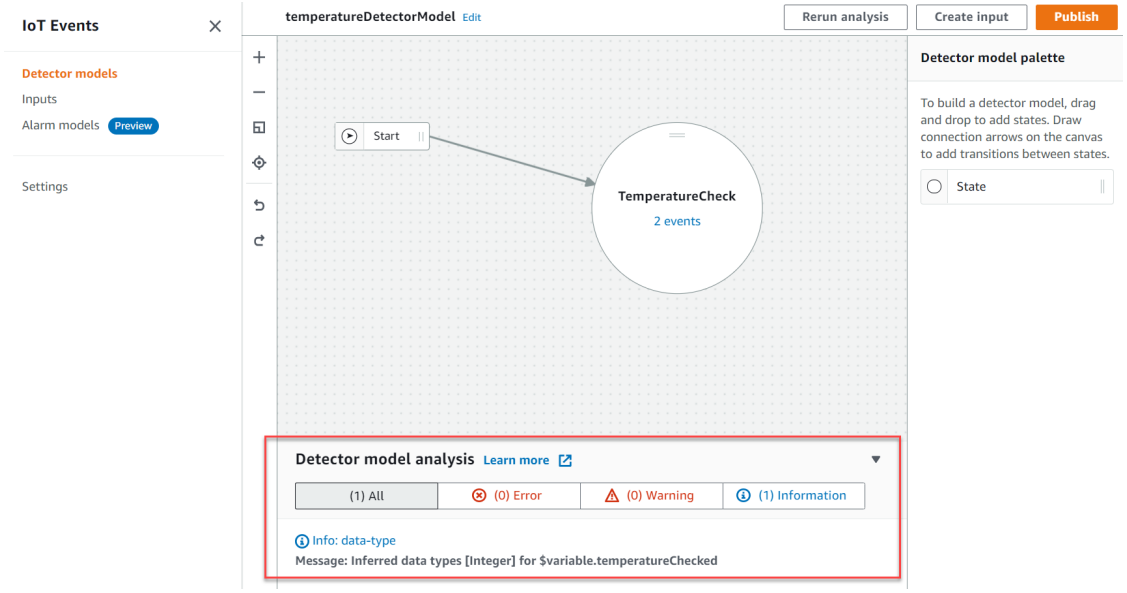
分析探测器模型（控制台）

以下步骤使用Amazon IoT Events控制台用于分析探测器模型。

1. 登录到 Amazon IoT Events 控制台。
2. 在导航窗格中，选择探测器模型.
3. 下面探测器模型，选择目标探测器型号。
4. 在探测器型号页面上，选择编辑.
5. 在右上角，选择运行分析.



下面是分析结果示例Amazon IoT Events控制台。



Note

晚于Amazon IoT Events开始分析您的探测器模型，您最多有 24 小时的时间来检索分析结果。

分析探测器模型 (Amazon CLI)

以下步骤使用Amazon CLI以分析探测器模型。

1. 运行以下命令以开始分析。

```
aws iotevents start-detector-model-analysis --cli-input-json file://file-name.json
```

Note

Replace（替换）####使用包含探测器模型定义的文件名。

Example 探测器模型定义

```
{
    "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"
    }
}
```

如果您将Amazon CLI分析现有的探测器模型，选择以下选项之一以检索探测器模型的定义：

- 如果要Amazon IoT Events控制台，执行以下操作：
  1. 在导航窗格中，选择探测器模型.
  2. 下面探测器模型，选择目标探测器型号。
  3. 选择导出探测器模型从操作下载探测器模型。探测器模型以 JSON 保存。
  4. 打开探测器模型 JSON 文件。
  5. 你只需要`detectorModelDefinition`object。删除以下内容：
     - 第一个大括号 ({) 在页面顶部
     - 这些区域有：`detectorModel`线

- 这些区域有：`detectorModelConfiguration`宾语
- 最后一个大括号 (}) 在页面底部

6. 保存该文件。

- 如果要Amazon CLI，执行以下操作：

1. 在终端中运行以下命令。

```
aws iotevents describe-detector-model --detector-model-name detector-model-name
```

2. Replace（替换）*detector-model-name*将名称与您的探测器模型一起使用。
3. 复制`detectorModelDefinition`向文本编辑器反对。
4. 添加大括号 ({}) 之外的源`detectorModelDefinition`.
5. 将该文件保存到JPON中。

Example 响应示例

```
{
    "analysisId": "c1133390-14e3-4204-9a66-31efd92a4fed"
}
```

2. 从输出中复制分析 ID。
3. 运行以下命令以检索分析的状态。

```
aws iotevents describe-detector-model-analysis --analysis-id "analysis-id"
```

Note

Replace（替换）*analysis-id*使用您复制的分析 ID。

Example 响应示例

```
{
    "status": "COMPLETE"
}
```

状态可以是以下值之一：

- `RUNNING`–Amazon IoT Events正在分析您的探测器模型。这一过程耗时最多 1 分钟。
- `COMPLETE`–Amazon IoT Events已完成对探测器模型的分析。
- `FAILED`–Amazon IoT Events无法分析您的探测器模型。请稍后重试。

4. 运行以下命令以检索探测器模型的一个或多个分析结果。

Note

Replace（替换）*analysis-id*使用您复制的分析 ID。

```
aws iotevents get-detector-model-analysis-results --analysis-id "analysis-id"
```

Example 响应示例

```
{
    "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

晚于Amazon IoT Events开始分析您的探测器模型，您最多有 24 小时的时间来检索分析结果。

Amazon IoT Events 命令

本章引导您完成所有的 API 操作Amazon IoT Events详细信息，包括支持的 Web 服务协议的请求、响应和错误示例。

Amazon IoT Events 操作

您可以使用Amazon IoT Events用于创建、读取、更新和删除输入和探测器模型以及列出其版本的 API 命令。有关更多信息，请参阅 。行动和数据类型支持的Amazon IoT Events在里面Amazon IoT EventsAPI 参考.

这些区域有：Amazon IoT Events部分在里面Amazon CLI命令参考包括Amazon CLI可以用来管理和操作的命令Amazon IoT Events.

Amazon IoT Events 数据

您可以使用Amazon IoT EventsData API 命令用于向探测器发送输入、列出探测器以及查看或更新探测器的状态。有关更多信息，请参阅 。行动和数据类型支持的Amazon IoT Events中的数据Amazon IoT EventsAPI 参考.

这些区域有：Amazon IoT Events数据节在里面Amazon CLI命令参考包括Amazon CLI你可以用来处理的命令Amazon IoT Events数据。

文档历史记录

下表介绍了Amazon IoT Events开发人员指南2020 年 9 月 17 日之后。如需对此文档的更新的更多信息，您可以订阅 RSS 源。

update-history-change	update-history-description	update-history-date
区域启动	Amazon IoT Events 现已在亚太地区 (孟买) 提供。	2021 年 9 月 30 日
区域启动	Amazon IoT Events现于中可用 Amazon GovCloud （US-West) 区域。	2021 年 9 月 22 日
通过运行分析对探测器模型进行故障排除	Amazon IoT Events现在可以分析您的探测器模型并生成可用于对探测器模型进行故障排除的分析结果。	2021 年 2 月 23 日
区域启动	启动Amazon IoT Events在中国（北京）。	2020 年 9 月 30 日
表达式用法	添加了向您展示如何编写表达式的示例。	2020 年 9 月 22 日
使用警报进行监控	警报可帮助您监控数据是否有变化。您可以创建警报，在超过阈值时发送通知。	2020 年 6 月 1 日

早期更新

下表介绍了Amazon IoT Events开发人员指南2020 年 9 月 18 日之前。

更改	说明	日期
加	添加了类型验证参考 (p. 101).	2020 年 8 月 3 日
加	增加区域信息使用其他Amazon服务 (p. 89).	2020 年 5 月 7 日
添加、更新	添加了负载自定义功能和新的事件操作：Amazon DynamoDBAmazon IoT SiteWise.	2020 年 4 月 27 日
编辑	添加了对状态机概念的新描述。内容的常规编辑。	2019 年 10 月 31 日
加	为探测器模型条件表达式添加了新的内置函数。	2019 年 9 月 10 日
加	添加了探测器模型示例。	2019 年 8 月 5 日

更改	说明	日期
加	添加了新的事件动作：Lambda、Amazon SQS、Kinesis Data Firehose 和Amazon IoT EventsInput。	2019 年 7 月 19 日
增补、更正	纠正了描述`timeout()`函数。添加了有关账户不活跃的最佳实践。	2019 年 6 月 11 日
纠正	更新：控制台调试选项页面图片；控制台权限策略。	2019 年 6 月 5 日
更新	Amazon IoT Events服务现已公开发布。	2019 年 5 月 30 日
添加、更新	更新了安全信息；添加了带注释的探测器模型示例。	2019 年 5 月 22 日
纠正	更新：限量预览版下载链接；Amazon SNS 负载示例；增加了所需权限 CreateDetectorModel.	2019 年 5 月 17 日
加	添加了有关安全的信息。	2019 年 5 月 9 日
纠正	限量预览版下载链接已更正。	2019 年 4 月 19 日
编辑	编辑改进。	2019 年 4 月 16 日
限量预览版	该文档的有限预览版。	2019 年 3 月 28 日
编辑	编辑改进。	2018 年 5 月 18 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。